OVERCOMING CUBESAT DOWNLINK LIMITS WITH VITAMIN: A NEW VARIABLE CODED MODULATION PROTOCOL

By

Thomas A. Sielicki

RECOMMENDED:

Dr. Jon Hamkins

Dr. Joseph Hawkins

Dr. Charles Mayer

Dr. Denise Thorsen
Advisory Committee Chair

Dr. Charles Mayer
Chair, Department of Electrical and Computer Engineering

APPROVED:

Dr. Douglas Goering
Dean, College of Engineering and Mines

Dr. John Eichelberger
Dean of the Graduate School

Date

OVERCOMING CUBESAT DOWNLINK LIMITS WITH VITAMIN: A NEW VARIABLE CODED MODULATION PROTOCOL

A

THESIS

Presented to the Faculty

of the University of Alaska Fairbanks

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

By

Thomas A. Sielicki, B.S.

Fairbanks, Alaska

December 2013

Abstract

Many space missions, including low earth orbit CubeSats, communicate in a highly dynamic environment because of variations in geometry, weather, and interference. At the same time, most missions communicate using fixed channel codes, modulations, and symbol rates, resulting in a constant data rate that does not adapt to the dynamic conditions.

When conditions are good, the fixed date rate can be far below the theoretical maximum, called the Shannon limit; when conditions are bad, the fixed data rate may not work at all. To move beyond these fixed communications and achieve higher total data volume from emerging high-tech instruments, this thesis investigates the use of error correcting codes and different modulations. Variable coded modulation (VCM) takes advantage of the dynamic link by transmitting more information when the signal-to-noise ratio (SNR) is high.

Likewise, VCM can throttle down the information rate when SNR is low without having to stop all communications. VCM outperforms fixed communications which can only operate at a fixed information rate as long as a certain signal threshold is met.

This thesis presents a new VCM protocol and tests its performance in both software and hardware simulations. The protocol is geared towards CubeSat downlinks as complexity is focused in the receiver, while the transmission operations are kept simple. This thesis explores bin-packing as a way to optimize the selection of VCM modes based on expected SNR levels over time. Working end-to-end simulations were created using MATLAB and LabVIEW, while the hardware simulations were done with software defined radios. Results show that a CubeSat using VCM communications will deliver twice the data throughput of a fixed communications system.

Table of Contents

Page

Page

List of Figures

Page

Page

List of Tables

Page

List of Appendices

Page

Acknowledgments

A portion of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA) and funded by the Alaska Space Grant Program (ASGP).

Chapter 1
Thesis Introduction

1.1 Significance of CubeSats

The current development of CubeSats has introduced the idea that small and simple satellites can be used for important scientific research with a relatively low cost for construction. A 1U CubeSat measures 10 cm x 10 cm x 10 cm in size. Other common sizes are 2U and 3U which are respectively double and triple in length. CubeSats are standardized by specifications developed by California Polytechnic State University in the late 1990's [1]. Using this standard multiple CubeSat units are able to be packaged into a Poly-Picosatellite Orbital Deployer (P-POD). The P-POD launches as an auxiliary payload in normally unused space on larger and more expensive space missions, reducing the cost to fly to about $40,000 per U [2].

With the cheap cost of CubeSat construction and deployment, there are obviously going to be limitations to overcome. CubeSats are size limited, providing less space for solar panels and large antenna systems, which creates significant limitations for the communications system as it operates at the low power levels, about 2 W for the 1U.

1.2 The Communications Link Budget

The signal power received at a ground station from a transmitting satellite is described by the basic link budget shown,

$$P_{RX} = P_{TX} + G_{TX} - L_{TX} - L_{FS} + G_{RX} - L_{RX} - L_M \,(\text{dB}). \tag{1.1}$$

The amount of power received (P_{RX}) depends on the transmitted power (P_{TX}), gain of the transmitter (G_{TX}), sum of the losses at the transmitter (L_{TX}), free space loss in a wireless communication channel due to distance (L_{FS}), the gain at the receiver (G_{RX}), the sum of the losses at the receiver (L_{RX}), and any other miscellaneous losses (L_M). Ultimately the goal in any communication system is to maximize the received power, P_{RX}, as this directly relates to achieving a higher information rate.

Looking closer at the received power, it is convenient to express P_{RX} in terms of signal-to-noise ratio (SNR), as described by

$$SNR = (E_s/N_0) \cdot (R_s/B), \tag{1.2}$$

where E_s is the energy per symbol, N_0 is the one-sided noise spectral density, R_s is the symbol rate, and B is the noise bandwidth. The channel is assumed to be well modeled by Additive White Gaussian Noise (AWGN). So in a communication environment with a constant noise spectral density, bandwidth, and symbol rate, only a higher signal to noise ratio can result in more channel capacity, C, as seen by Shannon's channel coding theorem [3]

$$C = B \log_2(1 + SNR) \text{ bits/s.} \tag{1.3}$$

Channel capacity is in high demand as scientific mission requirements are demanding more downlink throughput than ever before; for example high-tech space instruments like 50 megapixel resolution imagers [4]. Compounding the data downlink bottleneck, low Earth orbiting CubeSats can only communicate with a single ground station for a 15 minute duration every 90 minutes, at best. This in turn requires the communications engineer to focus on every aspect of the link budget to increase SNR and thus downlink throughput.

Current CubeSat missions have taken many different approaches to increase the daily downlink throughput. One, not so cheap, but non-satellite-impacting approach is increasing the receiver gain, G_{RX}, which in simplest terms is using a bigger receive antenna or satellite dish. The investment difference between a 3 meter ground antenna ($4,000) and a 9 meter antenna ($116,000) is $112,000, which is not practical considering a typical CubeSat budget [5]. Even the cheapest cost of renting downlink time can quickly exceed the CubeSat's overall pricetag. Likewise, G_{TX}, the gain of the transmitting antenna on the CubeSat is size limited on the tiny 10 cubic centimeter satellite.

Data throughput can also be increased without manipulating the link budget, but instead increasing the communication time. A globally distributed ground station network increases the communication window and can provide more constant communications, which increases downlink throughput but can also be very expensive [6].

Some designers have used adaptive data rates. Adaptive data rates are commonly used as an alternative to or in conjunction with variable coding and modulation. The idea is to simply send the bits faster when SNR is strong and slower when SNR is low. This technique typically requires a greater bandwidth (B) due to occasional high symbol rates which might not be available or affordable [7].

Looking at the link budget equation there are several loss terms. These are the losses which are mainly uncontrollable. L_{TX} and L_{RX} are the losses in the wiring and hardware of

the transmitter and receivers and are kept to a minimum in the satellite design. L_M accounts for the miscellaneous losses like polarization mismatch, implementation loss, fading, and atmospheric conditions. Polarity mismatch can be solved through circular patterns and controlled pointing of the spacecraft and on the ground.

This thesis looks at solving the communications data throughput issue by adapting to the expected variations in the free space loss, L_{FS}, or path loss margin from low orbit geometries with time-varying codes and modulations. This is a technique that can double throughput versus a traditional system with the low cost of increasing the system complexity and avoiding the high monetary cost. Path loss is defined as $L_{FS} = 20 \log_{10} \frac{4\pi d}{\lambda}$, where λ is the wavelength of the radio frequency. The path loss term in Equation 1.1 shows an inverse squared relationship with distance (d). Since the orbiting satellite is not at a constant distance from the ground station the free space loss, L_{FS}, in the link budget equation, is not a constant value but variable with a minimum and maximum. This margin results in a received power that changes over time, and a communication system that accounts for this will have the ability to obtain a higher channel throughput.

1.3 Forward Error Correcting Codes

Today's CubeSat missions typically communicate using fixed channel codes, modulations, and symbol rates, resulting in a constant information rate that does not adapt to the dynamic conditions. The information rate is the rate at which the scientific data is transmitted. When conditions are good, the fixed information rate can be far below the theoretical maximum; when conditions are bad, the fixed information rate may not work at all. CubeSat space missions communicate in a highly dynamic environment because of variations in geometry, weather, and interference. To move beyond these fixed communications modes and achieve higher total data volume this thesis investigates time-varying codes and modulations.

Variable Coded Modulation (VCM) takes advantage of the dynamic link by increasing the information rate when SNR is high, whereas fixed communications transmits the same information rate as long as a certain signal threshold is met. Likewise, VCM can throttle down the information rate when SNR is low without having to stop all communications. This allows a higher total throughput than can be achieved with fixed-mode communications. Additionally implementation of time-varying codes and modulations can be processed in the CubeSat size constraints through low power Field Programmable Gate Array (FPGA) [8], while the computationally complex operation of decoding would

be handled by the ground station.

VCM is an important topic for all space communication applications, for example probes sent into space experience time varying dynamic links as they orbit around other planets. NASA currently moves massive amounts of scientific data and images from the Mars Science Laboratory up to the low altitude Mars Reconnaissance Orbiter, where the communication window is short and dynamic just like CubeSats. The resources are extremely limited as there is literally nothing available on Mars in terms of communication resources, so even with billions of dollars, antenna size, power levels, ground stations, and satellite relays will remain hindrances.

The early advances of Forward Error Correction (FEC) brought forth the concept of VCM. In 1982, Gotfiend Ungerboeck showed how FEC (specifically trellis codes) could be applied to double the current speed of telephone modems using the same infrastructure and bandwidths. In 1994 the Digital Video Broadcasting - Satellite (DVB-S) standard was implemented for satellite video and data to be transmitted using different coding rates and modulation types, and in 2003 second generation technology Digital Video Broadcasting - Satellite Second Generation (DVB-S2) included Adaptive Coded Modulation (ACM) technologies [9]. These are just two examples of how coding and modulation have helped communications approach the Shannon limit.

Forward error correcting allows for the recovery of bit errors in noisy channels without the retransmission of data. This is done by encoding the information bits with redundancy. The simplest forward error correcting is simply repeating the bits, for example sending the message 010 as 000111000 via the (3,1) redundancy code. The problem with this simplest solution is noise occurs randomly in time and will not effect every bit equally, meaning a burst could result in receiving 0??????00. In this case, the middle bit is unknown, while the first bit has more uncertainty than then last. Efficient error correcting methods add redundancy bits that are complex functions of the information bits and encode the data in much longer lengths which results in averaging out the noise over time.

The Hamming (7,4) code which was developed by Richard Hamming in 1950 at Bell Laboratories is the first practical Forward Error Correction (FEC), in the field of information theory [10]. Although this thesis uses more advanced FECs, the principles are the same. The Hamming code takes in 4 information bits (the bits of data needing to be sent somewhere) and always sends 7 bits. The receiver is able to recover the message with 1 bit error, and it can detect 2 bit errors. This is explained visually on a Venn diagram (Figure 1.1), d_{1-4} are

the information bits while p_{1-3} are the extra parity bits. The parity bits become either 0 or 1 based on whichever value makes the circle sum to an even number. For example, if the three information bits, d_1, d_2, d_4, are all 1s, the parity bit, p_1, will be 1 for an even sum.

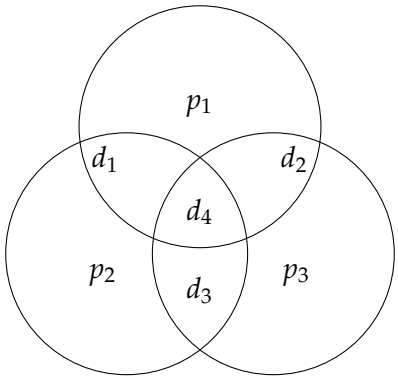

Figure 1.1. Hamming (7,4) Venn Diagram

Figure 1.2 shows an example of the transmitted and received bits in a Venn diagram form. The information bits of 1,0,0,1 are shown in blue and the parity bits 0,1,0 are in red. In this case the decoder sees that the bottom two circles do not meet their parity checks, and the only single bit flip that fixes this is changing d_3 (gray) from a 1 to a 0. The error correcting code successfully transmits the 4 information bits to the receiver in an environment where 6 or 7 (of 7) bits are successfully recovered. Not using the coding in the same conditions will result in corrupted data and require retransmission, which requires feedback communication, and in the case of a CubeSats additional time and power.

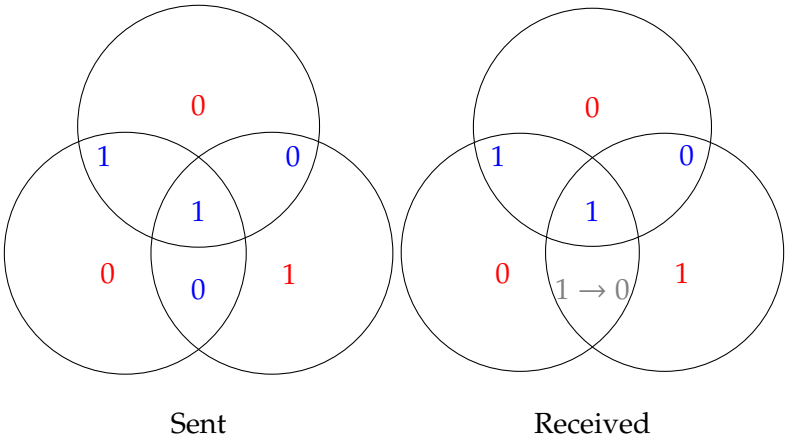

Figure 1.2. Example of Hamming (7,4)

Figure 1.3 shows the performances of the VCM modes investigated in this thesis and how close they come to achieving the Shannon limit (Equation 1.3) by nearly matching the curve [3]. Each marker represents a combination of one coding type set to a specific rate and modulation. Similar modulations are linked together by the same colored line. All 28 modes are described in Table 2.1. A traditional fixed communications system would only operate at one mode, which could be any of the 28 data points. The fixed mode system would fail to operate below a threshold and is easily outperformed above the threshold as it diverges from the Shannon Limit.

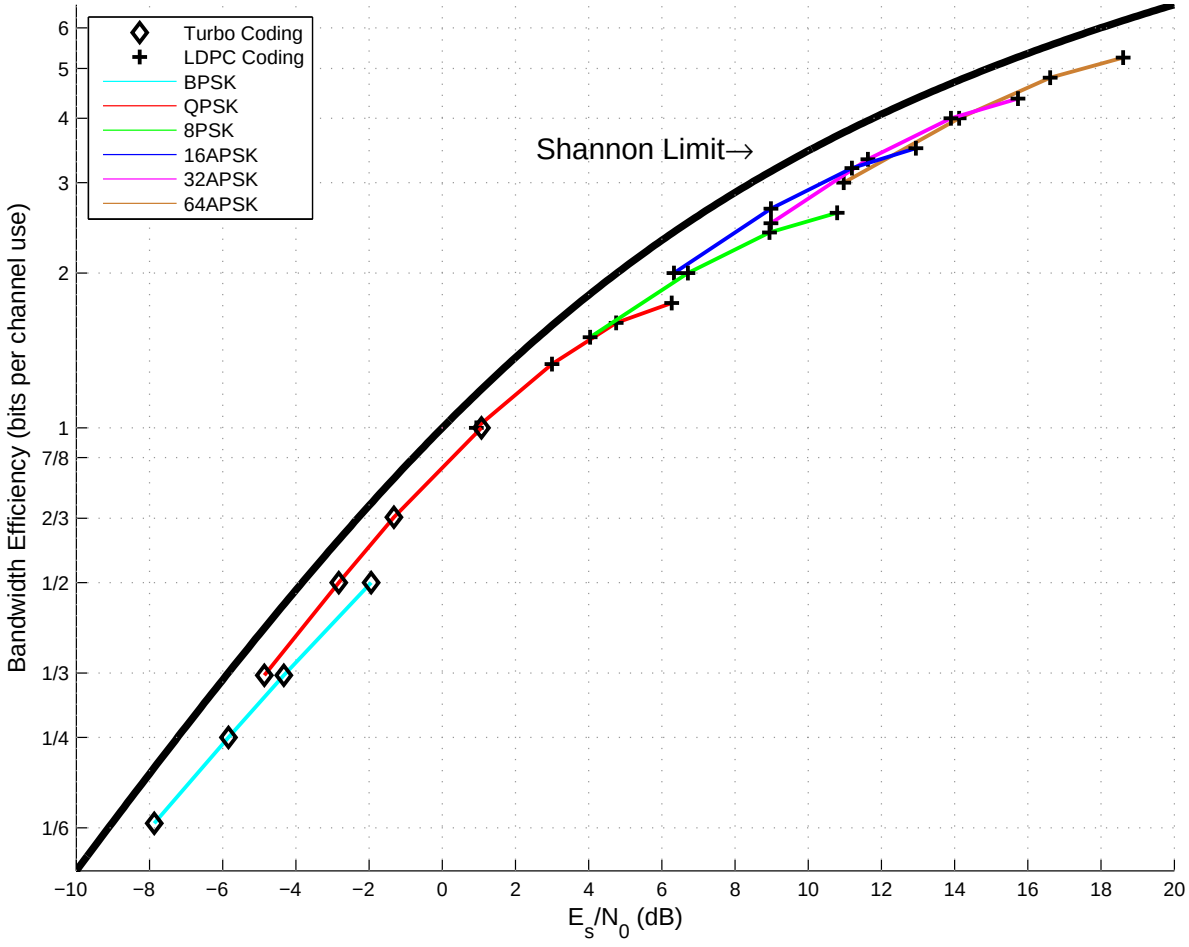

Figure 1.3. The VCM Modes vs Shannon Limit

Adaptive Coded Modulation (ACM) is an extension VCM, as it incorporates a protocol for determining the VCM mode to use. ACM requires a two way communication system that provides feedback of received signal levels in real time to transition between operating

modes. ACM has been applied in two way systems like cellular data communications since the late 1990's [11] and consumer satellite internet services like HughesNet. The major advantage of ACM is instantaneous adaption to the real time link budget, which can account for factors like weather disturbances, unexpected antenna pointing, and cochannel interference. VCM is a much simpler communication system which has some advantages and disadvantages. Needing to know the link budget and a detailed path loss before hand is not possible in most applications but it is with low earth orbiting CubeSats. Additionally, nearly all CubeSats have the mission of collecting data and transmitting to Earth, not the other way around, making a one way link practical.

1.4 Thesis Overview

The purpose of this thesis is to introduce a new VCM protocol, Variable Coded Modulation to Maximize Information (VITAMIN). The performance of the VITAMIN design will be tested and measured to quantify the data throughput improvement versus costs. The VITAMIN design utilizes the industry's best error correcting codes as recommended by NASA and is shown to be an improvement to other industry VCM systems, like Serial Concatened Convolutional Code (SCCC) from Consultative Committee for Space Data Systems (CCSDS) [12].

Chapter 2 is a proof of concept for applying VCM to CubeSat applications. An entire communications simulation was developed in software first to determine the feasibility of a new VCM communication system, which evolved into VITAMIN. This step was essential to identifying design obstacles and requirements before moving to a hardware product. Additionally this process provided the benchmark suggesting that downlink throughput can be nearly doubled with several VCM designs when applied to a CubeSat application.

Since VITAMIN is specifically proposed as a VCM system, not adaptive (ACM), a mode calculation program is required for operation. Additionally the system is made flexible to work in a range of symbol rates, link budgets (more specifically path loss ranges), and any subset of the VITAMIN operating modes. Chapter 3 discusses the research and development of a software program that predictably maximizes the information throughput for a satellite application. Algorithms similar to bin-packing are discussed and applied when looking at mode packing. Extensive statistical analysis is done to measure the improvements VITAMIN has over traditional and alternative communications approaches, plus information throughput is compared to the theoretical maximum.

Chapter 4 is an end-to-end hardware implementation of the VITAMIN protocol. Software defined radios were coded to implement both a VITAMIN transmitter and receiver. Chapter 4 goes through the information processing from the bits into the transmitter to the bits out of the receiver. Unlike Chapter 2, the protocol is tested using radio frequencies as the signal travels through the air or cable. Common communication issues like receiver syncing and overcoming noise are encountered. The receiver developed not only proves the proof of concept, but is also a functional ground station receiver.

Chapter 5 is the performance analysis of the transmitter and receiver developed in Chapter 4. Simulations include codeword error rate analysis, VCM mode switching, and real time satellite to ground communication.

Chapter 6 fully describes the developed VITAMIN protocol. As research and implementation created new ideas and resolved issues, the VITAMIN protocol changed. Chapter 6 describes the current state of the protocol and provides documentation on how to implement VITAMIN. The second part of Chapter 6 discusses future work that needs to be done with the VITAMIN protocol and work required to turn the software defined radio receiver into a reliable ground station receiver.

Chapter 2
Developing and Simulating the VCM Protocol

2.1 Introduction

Chapter 2 reports on the design and performance of a new VCM system. This VCM system comprises eight of NASAs recommended codes from the CCSDS standards [13], including four turbo and four Accumulate Repeat 4 Jagged Accumulate (AR4JA) low-density parity-check codes, together with six modulations types [14]. The signaling protocol for the transmission mode is based on another CCSDS recommendation [12]. The coded modulation may be dynamically chosen to optimize throughput.

The purpose of this chapter is to introduce a modified VCM system tailored for large data downlinks for low orbiting satellites, mainly CubeSats. This chapter investigates a selection of specific VCM modes and a new protocol for mode detection based solely on frame markers. This chapter is focused on a Matrix Laboratory (MATLAB) end-to-end simulation that consists of random data generation, encoding the data, modulation of the data, additive white Gaussian noise for channel simulation, frame marker identification, coded modulation mode extraction, demodulation, and decoding. Achievable error rates and total throughput are recorded. This simulation can be applied to any link budget that describes the SNR over time and can selectively use all or some of the supported coded modulation modes.

Lastly, a novel aspect of the VCM design is that each operating mode has a unique number of symbols in its frame. This enables the receiver to identify the code and modulation simply from the number of symbols occurring between the dedicated frame markers which occur between frames, without having to explicitly transmit a signal to identify the operating mode. This concept allows for a limitless selection of modes and low complexity design.

2.2 The VITAMIN System Design

The proposed VCM for CubeSat application, VITAMIN, closely follows the VCM protocol used in the CCSDS recommendation for Serial Concatened Convolutional Code (SCCC) [12]. Namely, a Physical Layer Frame (PLF) consists of a Frame Marker (FM), a Frame Descriptor (FD), and 16 frames of the same VCM type, shown in Figure 2.1. The PLF symbol length varies based on the VCM mode being transmitted. Attached sync markers are inserted into the data and the input goes through a pseudo-random interleaver

prior to encoding. The FM is generated using a gold sequence described in the CCSDS recommendation [12]. This FM is 256 bits long and is modulated with Binary Phase Shift Keying (BPSK). The FM always follows the last or 16th codeword and precedes the FD, which signals which code and modulation will be used in the remainder of the Physical Layer Frame (PLF). The FD is generated as section 5.3.3.2.2 of the CCSDS recommended standard [12] suggests. The FD has a total length of 64 bits, can describe up to 32 VCM modes, and is also modulated with BPSK.

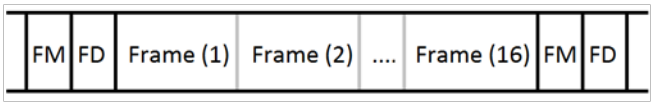

Figure 2.1. Physical Layer Frame [12]

This chapter shows that the VCM protocol of CCSDS can be used with two different FEC code sets, turbo and Low-Density Parity-Check (LDPC) codes [13]. Both of which have been selected for implementation in the VITAMIN system. It also shows that the existing VCM protocol of CCSDS is general enough to support a variety of codes and modulations [12].

2.2.1 CCSDS Turbo and LDPC codes

CCSDS has recommended a set of 16 turbo codes [13]. There is a code for each of rates 1/6, 1/4, 1/3, and 1/2, and input lengths 1784, 3568, 7136, and 8920. These codes have been a workhorse for deep space missions, for example, having been used on the Mars Reconnaissance Orbiter and the Curiosity Mars Rover. From this set of 16 codes, this thesis will use the turbo codes of length 8920 and rates 1/6, 1/4, 1/3, and 1/2.

CCSDS has also recommended a set of nine Accumulate Repeat 4 Jagged Accumulate (AR4JA) LDPC codes, along with one other higher rate LDPC code [13]. The nine AR4JA codes represent each combination of code rates 1/2, 2/3, and 4/5 and input lengths 1024, 4096, and 16384. The 10th LDPC code, called C2, has input length 7136 and rate 223/255, or approximately 7/8. From this set of ten codes, this thesis will use the AR4JA codes of length 16384 and rates 1/2, 2/3, and 4/5, along with the C2 LDPC code [14] [13].

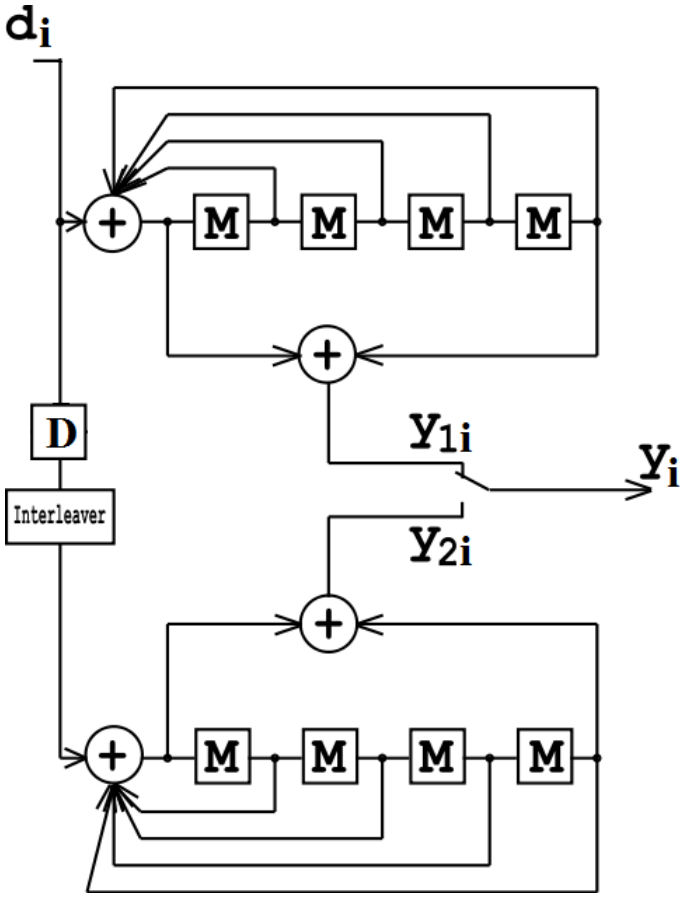

Figure 2.2. Turbo Encoder Processing Chain [15]

Both turbo and LDPC codes implement Forward Error Correction (FEC) algorithms. Figure 2.2 shows the general encoding process used in Turbo codes, where M represents a memory register and D represents a time delay. The turbo codes rely on parallel encoders that use a concatenation scheme to propagate the k input bits into n output bits. Every FEC has a code rate which is defined as k/n and typically expressed as a fraction. The decoder takes soft demodulation outputs and uses feedback registers to make bit decisions as the data is decoded. Soft demodulation is discussed in subsection 4.6.9. Only turbo codes of rates ranging 1/6 to 1/2 are used in this thesis.

The principle behind LDPC codes is parity nodes that are the modulo two sum of bit nodes. These bit and parity nodes are all interconnected by modulo two equations, thus the need to iterate to converge on a solution for the whole frame. For the AR4JA LDPC codes used in this thesis the frame length is 16384 bits, Figure 2.3 is a visual example showing

only a 6 bit chunk and the corresponding parity check nodes.

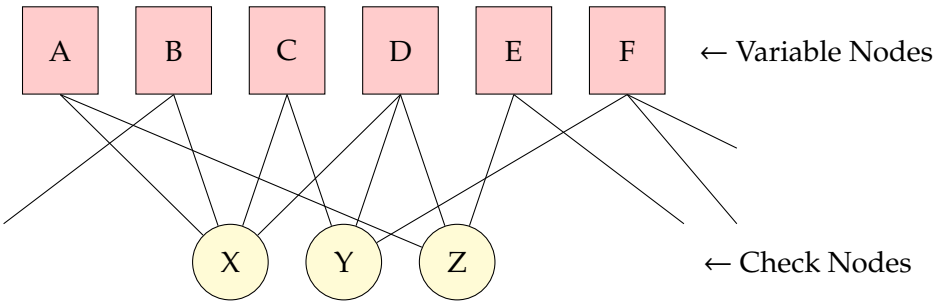

Figure 2.3. LDPC Example

Ignoring the inputs from other bits and just looking at the 3 check nodes X, Y, and Z the equations are:

$$\begin{aligned} X = A \oplus B \oplus C \oplus D = 0 \\ Y = C \oplus D \oplus F = 0 \\ Z = A \oplus D \oplus E = 0. \end{aligned} \tag{2.1}$$

Figure 2.4 replaces A-F with bit values of 1,0,0,1,0,1. Let's assume 3 bits are perfectly recovered and 3 bits are unknown (erased). In reality the LDPC decoder uses soft decision bits called log likelihood ratios, to represent the accuracy of each bit estimate. Looking at Figure 2.4, both Z and X can not accurately be computed because their inputs have unknowns. Fortunately Y can be calculated as D and F are both 1, meaning we are confident C must be 0 because Y must equal 0. This iteration is shown in blue. Now the green iteration process has more likely input for C which will result in a decision for A. Lastly red shows the propagation of confident values into E.

When the received values are noisy, multiple iterations happen until a successful solution is found, for example a more realistic input to the receive would be soft inputs A-F of 0.6,0.1,0.4,0.9,0.1, and 0.8 respectively. Variable and check nodes will recalculate their new values on each iteration based on combination of all previous estimates. Eventually a result will converge, as in the case of the first example where the received bits had high SNR, sometimes a solution will not converge resulting in an invalid solution that will have bit errors. More detailed introduction to LDPC coding is available in [16] [17].

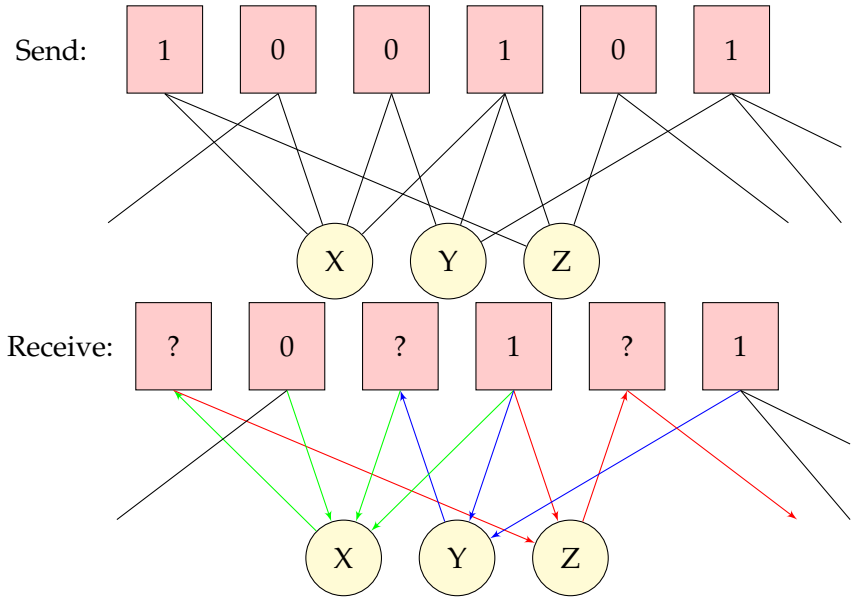

Figure 2.4. LDPC Example with Data

2.2.2 CCSDS Modulations

The CCSDS recommendation on SCCC [12] makes use of a number of modulations. Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), and 8 Phase Shift Keying (8PSK) which are text book modulations. The union of two Phase Shift Keying (PSK) constellations is used to form 16 Amplitude and Phase Shift Keying (16APSK). Two other modulations, 32 Amplitude and Phase Shift Keying (32APSK) and 64 Amplitude and Phase Shift Keying (64APSK), are also formed from the union of PSK constellations. CCSDS has described particular symbol mappings and relative amplitude scalings for these modulations [12], and the optimal demodulation of the modulations are described in [18].

2.2.3 The VCM modes

The VCM system described in this chapter uses 8 turbo code modes and 20 LDPC code modes. Additionally, all six modulation types discussed above are used. The 28 modes are listed in Table 2.1, with the shaded modes ultimately omitted from VITAMIN design as described below. The table gives the code rate, the number of information bits per modulation symbol (equal to the code rate times the base-2 logarithm of the number of constellation points of the modulation), the number of modulation symbols per physical layer frame, and the minimum E_s/N_0, in dB. To generate PLFs to meet the dynamic condi-

Table 2.1. Original 28 VCM Modes Investigated

	Modulation	Code	Code Rate	Info Bits per Symbol	PLF Length	E_s/N_0 for CWER 10^{-4}
0	BPSK	Turbo	1/6	0.17	857024	-7.87
1	BPSK	Turbo	1/4	0.25	571456	-5.84
2	BPSK	Turbo	1/3	0.33	428672	-4.33
3	BPSK	Turbo	1/2	0.50	285888	-1.94
4	QPSK	Turbo	1/6	0.33	428672	-4.86
5	QPSK	Turbo	1/4	0.50	285888	-2.83
6	QPSK	Turbo	1/3	0.67	214496	-1.32
7	QPSK	Turbo	1/2	1.00	143104	1.07
8	QPSK	AR4JA LDPC	1/2	1.00	262464	0.93
9	QPSK	AR4JA LDPC	2/3	1.33	196928	3.00
10	QPSK	AR4JA LDPC	4/5	1.60	164160	4.75
11	QPSK	C2 LDPC	7/8	1.75	65600	6.27
12	8PSK	AR4JA LDPC	1/2	1.50	175083	4.04
13	8PSK	AR4JA LDPC	2/3	2.00	131392	6.71
14	8PSK	AR4JA LDPC	4/5	2.40	109547	8.94
15	8PSK	C2 LDPC	7/8	2.62	43840	10.79
16	16APSK	AR4JA LDPC	1/2	2.00	131392	6.33
17	16APSK	AR4JA LDPC	2/3	2.67	98624	8.98
18	16APSK	AR4JA LDPC	4/5	3.20	82240	11.19
19	16APSK	C2 LDPC	7/8	3.50	32960	12.94
20	32APSK	AR4JA LDPC	1/2	2.50	105178	8.98
21	32APSK	AR4JA LDPC	2/3	3.33	78964	11.63
22	32APSK	AR4JA LDPC	4/5	4.00	65856	13.90
23	32APSK	C2 LDPC	7/8	4.37	26432	15.73
24	64APSK	AR4JA LDPC	1/2	3.00	87702	10.97
25	64APSK	AR4JA LDPC	2/3	4.00	65856	14.12
26	64APSK	AR4JA LDPC	4/5	4.80	54934	16.61
27	64APSK	C2 LDPC	7/8	5.25	22080	18.60

tions of the link, the minimum E_s/N_0 for each mode type must be known. Table 2.1 shows each mode's minimum E_s/N_0 needed for a Codeword Error Rate (CWER) of 10^{-4}. The VCM mode performances were analyzed in a previous paper [18].

2.3 CubeSat Implementation Feasibility

Only the downlink for CubeSat communications is power limited, resulting in the VCM system being applied for only outbound communications, which is a major key to the implementation of a VCM system in these small satellites. The computational complexity of the error correcting codes is found in the receiver which will be at the ground station.

The encoders are relatively low-complexity and can easily fit on a Field Programmable Gate Array (FPGA) like the Altera Cyclone or Xilinx Spartan3A which are already being included in other CubeSat designs [8].

2.4 Simulation

An end-to-end software simulation of the VITAMIN system was created. The software controlling the overall simulation was done in MATLAB, and the encoders and decoders for both LDPC and Turbo were implemented in C, for more efficient processing. MATLAB Executable (MEX) files were used to interface with the C encoders and decoders.

Several simulations of the communication systems were performed in MATLAB. The first analysis looked at the performances of the Frame Marker (FM) and Frame Descriptor (FD) under Additive White Gaussian Noise (AWGN). This was done to determine their reliability before implementation of the whole protocol. The simulation was performed millions of times, recording incorrect Frame Marker locations. Likewise incorrect FDs were recorded.

Next, each operating mode was tested. The entire process is visualized in Figure 2.5. One PLF was created by generating 16 frames of random data that were put through an interleaver and then encoded by a mode type. Next the bits were modulated to IQ pairs. The channel transmission is simulated with AWGN as described by a hypothetical CubeSat profile which describes SNR over time, this data is included in supplemental files discussed in Appendix A. Then the receiver part of the simulation occurred, the receiver first searched for the frame marker, decoded the frame descriptor that follows, and determined the VCM mode. Then the IQ pairs are demodulated and decoded. At last the simulation does a bit by bit comparison of the information sent and received recording any bit errors. This process was repeated until enough bits were processed to obtain measurable bit error rates.

The last MATLAB analysis consisted of a complete end-to-end simulation of a satellite pass. It was necessary to examine a hypothetical CubeSat profile with one ground station to determine the path loss over time. Knowing this information allowed for the determination of mode ordering and timing. The very first iteration of this was done by hand, and was time consuming. The next two satellite profile analyses were done slightly quicker with Microsoft Excel, but it became apparent a computer program for mode determination was needed. Hence the Throughput Mode Selection (TMS) program was created. This is discussed in Chapter 3. The end-to-end simulation processing was identical to the bit error

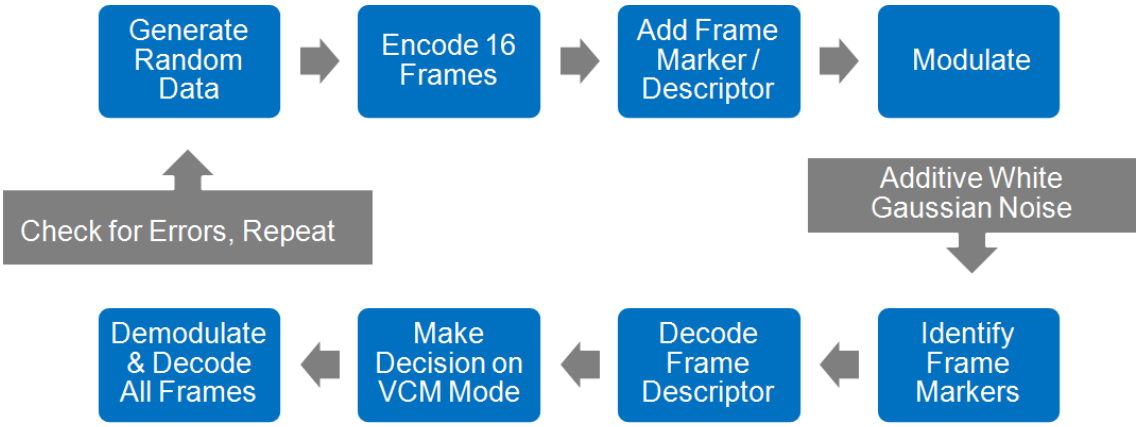

Figure 2.5. MATLAB Simulation Flow Chart

simulation, but with the modes determined.

2.5 System Performance

2.5.1 Frame Marker Identification Error Rate

A MATLAB simulation was performed to discover the error rate for detecting frame markers under Additive White Gaussian Noise (AWGN). This process consisted of a cross correlation between the noisy received signal and the expected 256 bit frame marker. The estimated offset index is given by:

$$\hat{x} = \underset{x:0\leq x\leq l-1}{\operatorname{argmax}} \sum_{k=0}^{255} s^*[k+x] \cdot m[k], \tag{2.2}$$

where s is the noisy signal, m is the known 256 bit frame marker, and l is length of the cross correlation or the length of the search. The search length l can be selected based on the length of the longest physical layer frame, among all VCM modes. Under the VCM design the longest physical layer frame possible is 857,024 symbols or bits since the modulation is BPSK. Doubling this length and subtracting 256 (the length of a frame marker) results in the length of the longest possible bit stream containing only one frame marker, which is 1,713,792 bits. This simulation generated two physical layer frames, chopped off the first frame marker, applied AWGN, and then computed the cross correlation to find the index with the highest correlation. If this index didn't match the known value, an error was recorded. This simulation was repeated one million times for each E_s/N_0 level of -20 to -8

dB. Fifty million simulations were needed for E_s/N_0 of -7.87 dB. The simulation results can be seen on Figure 2.6.

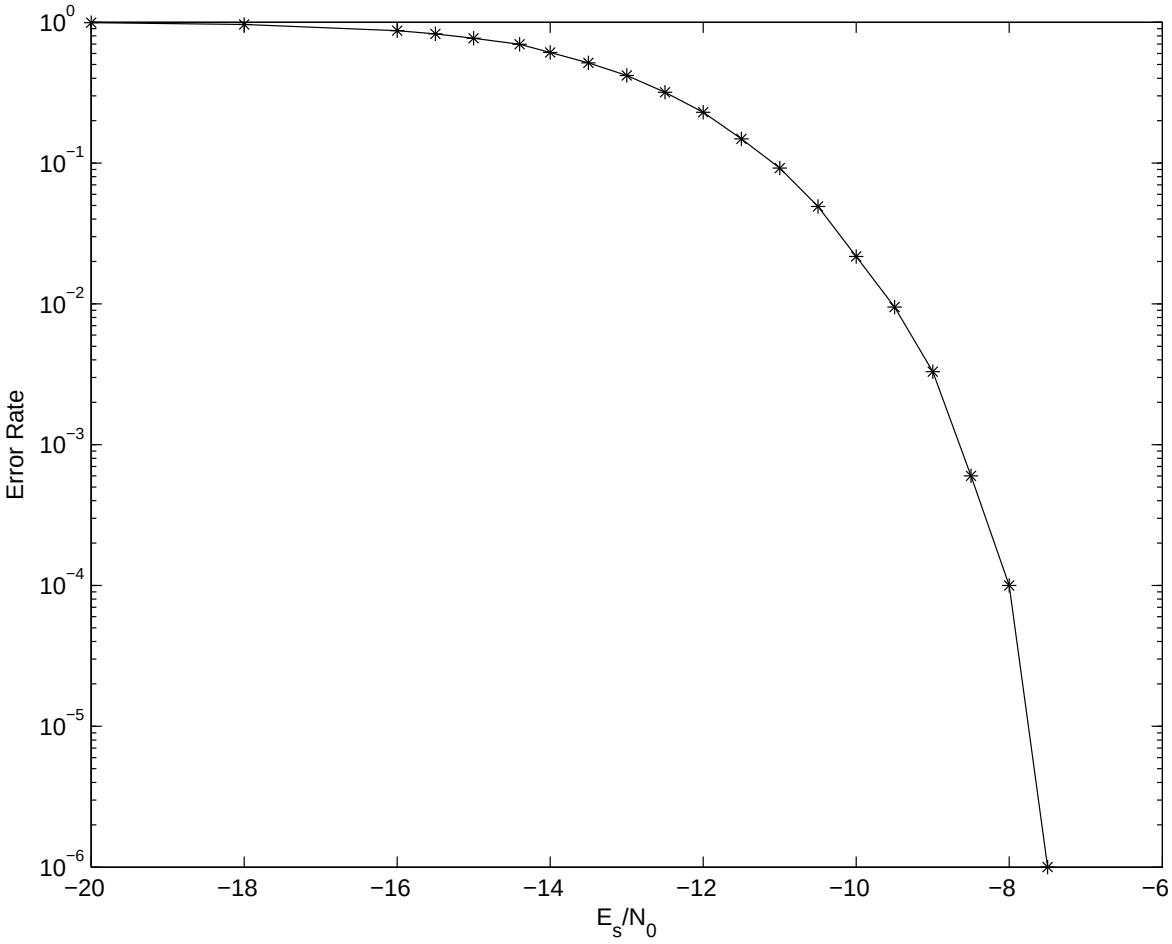

Figure 2.6. Frame Marker Identification Error Rate under AWGN

At an E_s/N_0 of -8 dB the Frame Marker is misidentified 1 in 10,000 trials, which is an acceptable level for even the lowest rate turbo code operating at E_s/N_0 of -7.87 dB with its own codeword error rate of 10^{-4}. This simulation shows that the CCSDS designed Frame Marker is acceptable for all codes considered in this VCM design.

2.5.2 Frame Descriptor Error

The next simulation was to test the Frame Descriptor performance under the same noisy conditions to ensure that the receiver will demodulate and decode using the right mode. This simulation assumed that the Frame Marker was correctly detected. The simulation

randomly selected one of the 32 frame descriptors, simulated a noisy channel with AWGN, decoded the noisy received frame descriptor, and checked if it was correctly detected. This simulation was then repeated 1 million times for six E_s/N_0 levels, -9 to -4 dB, which are the noisiest conditions expected under operation of this VCM system.

The receiver demodulation and decoding process of the frame descriptor utilized a lookup table of all 32 available frame descriptors. The FD of [12] uses a (32,6) code which is designed for a maximum of 32 modes where 5 bits are the identifier and 1 bit is reserved for a distributed pilot feature. This identifier is encoded to 32 bits and doubled to 64 bits for transmission. All 32 possibilities were tested even though the VCM system would use less. The FDs are stored in a 32 by 64 matrix in BPSK format of ±1, with $d_j[i]$ denoting the i^{th} bit in the Frame Descriptor for the j^{th} VCM mode. Finding the largest inner-product between the noisy input signal and all 32 frame descriptors yields the frame descriptor with the minimum distance to the received frame descriptor,

$$\hat{mode} = \underset{j:0\leq j\leq 31}{\operatorname{argmax}} \sum_{i=1}^{64} s[i] \cdot d_j[i]. \tag{2.3}$$

Figure 2.7 shows unacceptable error rates for frame descriptor identification for the same low signal levels that the turbo codes are designed to work. This means that the frame descriptor code will not work with the current VCM design. The simulation shows that when operating at E_s/N_0 of -7 dB the frame descriptor would be misidentified 25% of the time, and this would prevent reliable decoding of Turbo 1/6 QPSK codes that are able to operate down to -7.87 dB.

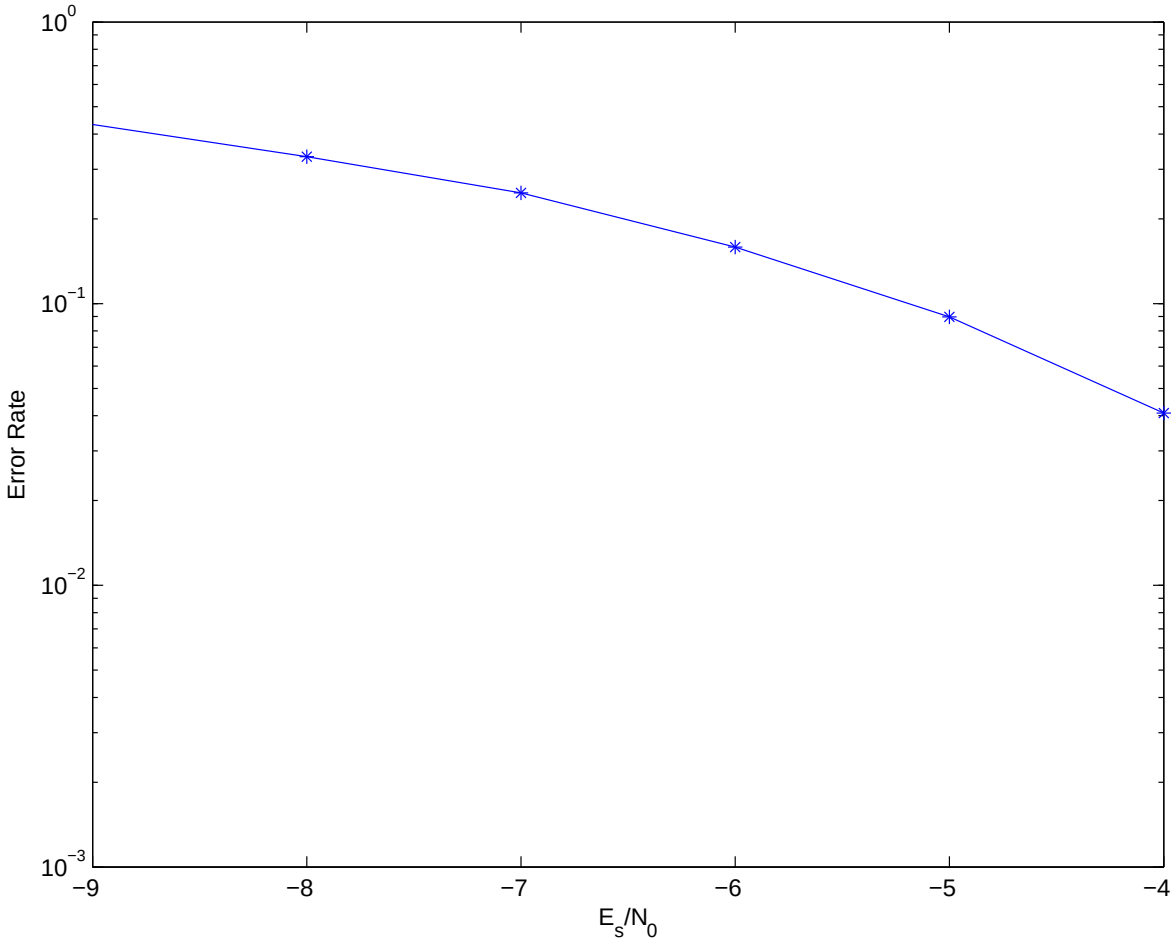

Figure 2.7. Frame Descriptor Identification Error Rate under AWGN

There are several solutions to this problem, the obvious choice is to design a new frame descriptor that has better error correcting capabilities. In theory, a (64,6) length code might accomplish this. Alternatively, the problem could be solved with a more complex receiver. When the receiver demodulates and decodes the received signal with the wrong demodulator and/or decoder, it can be discovered quickly by the failure of the decoder to converge to a valid codeword, or by the simple fact that the next frame marker does not appear in the expected location.

2.6 Design for Receiver Frame Marker Synchronization

Instead of reprocessing data with another mode type or decoding in parallel, this software simulation receiver was designed to find two frame markers, identify the number of sym-

bols between the markers, and deduce the mode from the length. This is a low-complexity, high-performing solution, and one novel result of the thesis.

This approach relies on the unique symbol length property. Of the 28 modes selected to be explored, only 24 were implemented, as the original 28 modes had all but four unique symbol lengths. Removing four codes would give 24 unique lengths, and as it turned out the four modes removed do not sacrifice VCM performance. An additional constraint to preserving unique lengths requires a constant symbol rate. This method completely eliminates the need for the receiver to process the poorly performing frame descriptor, suggesting that the VCM design could eliminate it all together. The final VITAMIN protocol adopts the design of using unique lengths and eliminating the Frame descriptor.

There are four pairs of modes that shared identical symbol lengths, they are listed below:

- mode 2 and 4 (BPSK 1/3 Turbo, QPSK 1/6 Turbo)
- mode 3 and 5 (BPSK 1/2 Turbo, QPSK 1/4 Turbo)
- mode 13 and 16 (8PSK 2/3 LDPC, 16APSK 1/2 LDPC)
- mode 25 and 22 (64APSK 2/3 LDPC, 32APSK 4/5 LDPC)

Looking at the eight modes, the solution of eliminating the usage of four modes can conveniently be justified. All BPSK modes serve little purpose since QPSK can deliver twice the information at no extra cost, so BPSK 1/2 and BPSK 1/3 can be removed (labeled Modes 2 & 3). Mode 13 can be eliminated because it is dominated by mode 16: mode 13 achieves a throughput of 2 bits per symbol and requires $E_s/N_0 = 6.71$ dB, while mode 16 achieves 2 bits per symbol and only requires $E_s/N_0 = 6.33$ dB. Similarly LDPC 32APSK 4/5 & LDPC 64APSK 2/3 both deliver 4 information bits per symbol but 64APSK 2/3 requires a higher signal to noise ratio to operate, so 64APSK 2/3 can be removed from the list of VCM modes. Although we have made these particular choices here, the available support for higher order modulations, and their relative implementation losses, will dictate which modes are dominated and can be eliminated in a particular system. For the purpose of this thesis, we assume support of all six modulations and equal implementation losses for all modulation types.

After reducing the VCM operating mode list to 24 unique modes, the frame descriptor could optionally be eliminated resulting in only frame markers separating the modes.

Additionally this new design allows for a limitless number of VCM modes which is an improvement over the 32 modes the frame descriptor method provided. Of course, any additional mode added will be required to have a unique symbol length, or some other means to identify it.

2.7 Overall Data Throughput Performance

Low Earth orbiting satellites are great candidates for VCM communications with primary ground stations, due to a varying link geometry and a fixed power budget. This section looks at the performance of using two alternative sets of VCM codes for a CubeSat mission, in addition to the Turbo and LDPC codes used in VITAMIN. The simulation input requires a link budget that describes the E_s/N_0 versus time for the duration of the pass. For this research Satellite Tool Kit software by Analytical Graphics Inc. was used to describe the link budget of a polar orbiting CubeSat at 600 km with a primary ground station in Fairbanks, AK. Two passes were generated with constant parameters for antenna gain and transmit power levels, with the only difference being the pass's maximum elevation. Two pass types were analyzed, a near overhead pass at 80 degrees maximum elevation and a slightly shorter pass with only 35 degrees maximum elevation. The 80 degree pass exhibits a minimum E_s/N_0 of -2.76 dB at the horizon and a maximum of 11.93 dB, with a total time of 635 seconds. The 35 degree pass exhibits a minimum E_s/N_0 of -2.79 dB at the horizon and a maximum of 7.89 dB, with a slightly shorter time of 591 seconds.

The two sets of pass data was then simulated with three sets of VCM codes: The 24 VCM modes from Table 2.1 labeled VITAMIN, Digital Video Broadcasting Satellite Second Generation for single carrier per transponder labeled DVB-S2 [19], and the SCCC encoder codes, labeled CCSDS SCCC [12]. VCM mode switching thresholds were calculated using each mode's respective minimum E_s/N_0 for a codeword error rate of 10^{-4} [18].

Additionally the simulation calculates the maximum fixed throughput using only one of the VCM modes. VCM improvement is calculated by dividing the VCM throughput by the fixed mode throughput. Table 2.2 shows the advantage of using VCM for these CubeSat passes and the comparison between all three VCM coding sets. This simulation uses a fixed symbol rate of 9600 kilo symbols per second.

Table 2.3. VCM Modes Used in 80 Degree Pass

Mode	Modulation	Code	Code Rate	Time Mode Used (s)	Percentage of Pass	Throughput (Kbits)
4	QPSK	Turbo	1/6	9	1.4	493
5	QPSK	Turbo	1/4	103	16.2	819
6	QPSK	Turbo	1/3	128	20.2	882
8	QPSK	AR4JA LDPC	1/2	92	14.5	512
9	QPSK	AR4JA LDPC	2/3	40	6.3	345
10	QPSK	AR4JA LDPC	4/5	50	7.9	768
12	8PSK	AR4JA LDPC	1/2	24	3.8	1420
16	16APSK	AR4JA LDPC	1/2	74	11.7	1689
17	16APSK	AR4JA LDPC	2/3	66	10.4	553
18	16APSK	AR4JA LDPC	4/5	18	2.8	992
21	32APSK	AR4JA LDPC	2/3	31	4.9	28

Table 2.2. Throughput of Different Coding Sets for Fixed and VCM

Pass Elevation	Pass Time (s)	Code Set	Fixed (Mbits)	VCM (Mbits)	VCM/Fixed
35	591	VITAMIN	3,437	6,564	1.91
35	591	DVB-S2	2,909	5,482	1.88
35	591	CCSDS SCCC	3,315	5,732	1.73
80	635	VITAMIN	3,878	8,502	2.19
80	635	DVB-S2	3,657	7,068	1.93
80	635	CCSDS SCCC	3,704	7,722	2.08

The simulations showed that all three sets of codes double or nearly double the overall throughput. The VCM design is able to take advantage of the 80 degree pass's higher variation in E_s/N_0 and exhibit more improvement. Additionally the VITAMIN set of codes slightly outperform CCSDS SCCC and DVB-S2 in total throughput. This can be attributed to the turbo codes which are able to operate when the link is the poorest, whereas both CCSDS SCCC and DVB-S2 are unable to utilize the link at all during the times the satellite is on the horizon.

Doubling the data throughput is a significant accomplishment for VCM and presents evidence that this system will be a valued asset in space communications. Figure 2.8 and Figure 2.9 are visual comparisons of fixed mode communications versus VCM showing the 35 and 80 degree passes, respectively. The solid curved line shows the actual link budget. The stepped blue line shows the E_s/N_0 thresholds needed for VCM communication. The dashed black line shows the best choice fixed mode E_s/N_0 threshold.

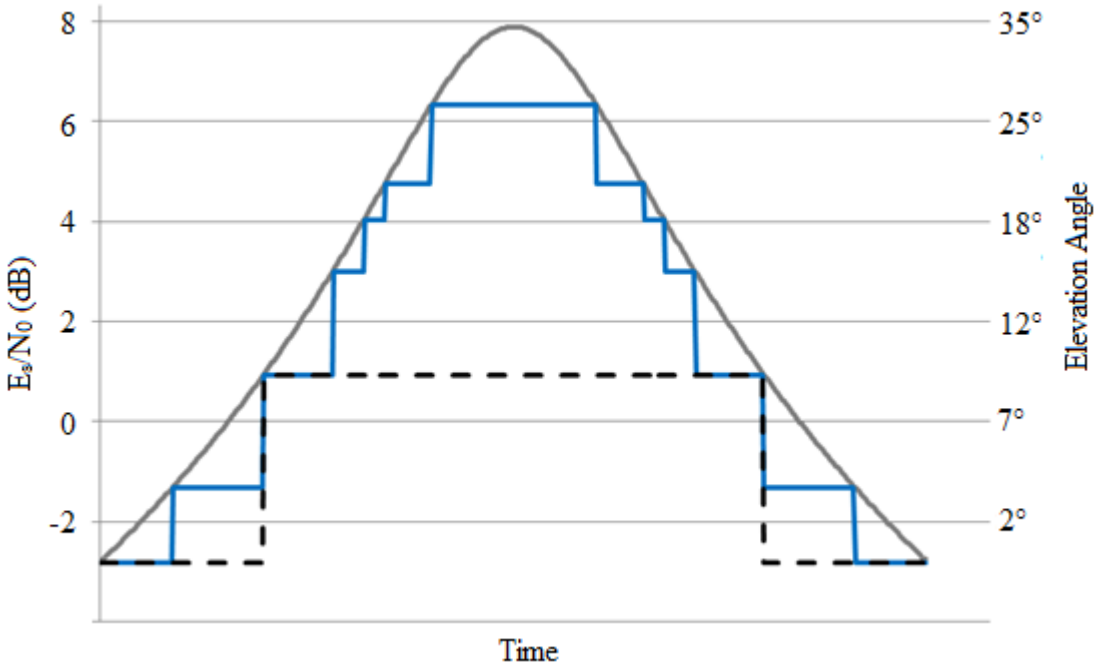

Figure 2.8. VCM versus Fixed Mode Communications for CubeSat Pass of 35 Degrees

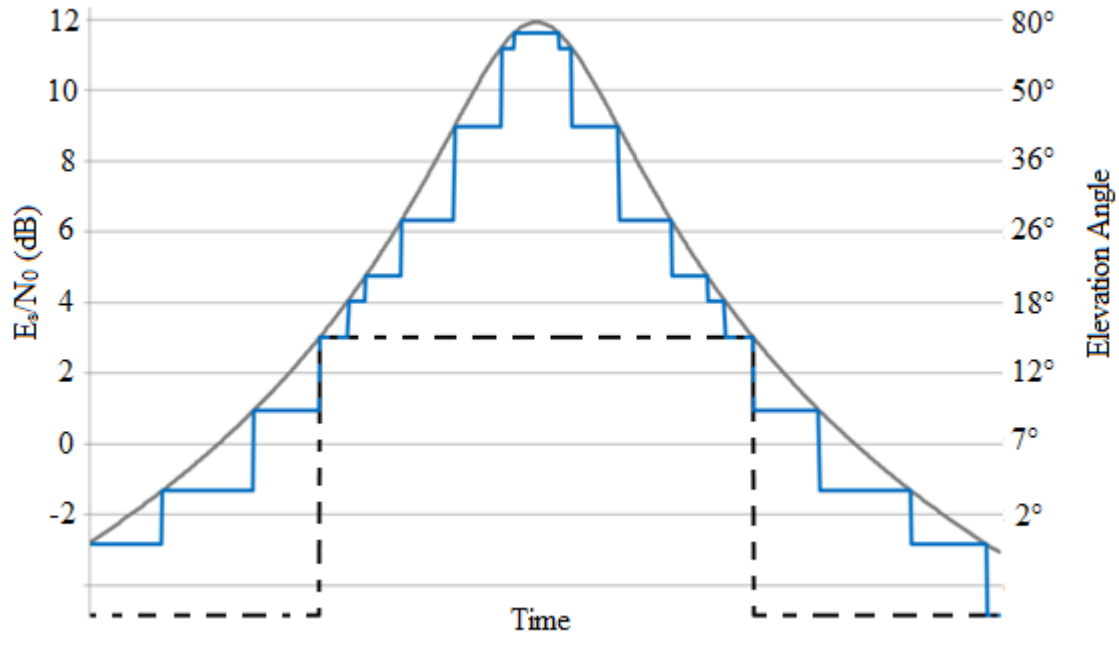

Figure 2.9. VCM versus Fixed Mode Communications for CubeSat Pass of 80 Degrees

For the case of the 80 degree maximum pass the best fixed mode communications system would be mode 9, AR4JA LDPC QPSK 2/3. Table 2.3 highlights that Mode 9 would provide the most throughput for that pass.

2.8 Low Symbol Rate Performance

There is a potential penalty associated with selecting certain VCM modes, if their associated physical layer frames are so long that conditions can become more favorable well before one full frame can be transmitted. This is evident in the case of CubeSats using error correcting codes at low symbol rates like 9600 symbols per second. This rate only yields 5,760,000 symbols per 10 minute pass. If the first VCM mode is QPSK Turbo 1/6 the first physical layer frame would be 428,672 symbols or 45 seconds.

The simplest VCM switching algorithm is selecting the mode with the highest information bit rate that can operate at both the current E_s/N_0 level and throughout the duration of its physical layer frame. If the link is varying quicker than the time it takes to transmit a physical layer frame, smarter decisions can be made when selecting VCM modes. Since there are 24 VCM modes with different lengths, information rates, and E_s/N_0 thresholds, all factors need to be considered when optimizing the highest throughput. Intelligently determining the VCM mode will be investigated in Chapter 3. This will help communications systems using VCM at low symbol rates and ensure proper mode transitioning for all noise levels.

Elements of Chapter 2 are published in IEEE Aerospace 2013, [20].

Chapter 3
Bin-Packing: Maximizing Throughput

3.1 Introduction

To successfully and efficiently implement the VITAMIN protocol for a satellite system, planning of the mode selection for each communication window needs to be done. All satellites have unique communication hardware and orbits and different ground stations, which means very different link budgets, path loss margins, and length of passes. Certain applications of the VITAMIN protocol could have very long pass times, others might have a short window for communication but a highly dynamic change in path loss. Additionally non-power-limited satellites can transmit at high symbol rates, while CubeSats transmit at very low symbol rates. Whatever the case might be, knowing all the communications parameters coupled with successful planning the VITMAIN implementation will yield a maximum information throughput. This chapter looks at the throughput maximization problem from the bin-packing approach and provides a flexible computer program that automatically preforms the VITAMIN mode selection.

Bin-packing has been an interesting and difficult problem for mathematicians and computer scientists for the last few decades. Bin-packing can be seen in 1, 2, or 3 dimensions. Bin-packing is simply minimizing wasted space by efficiently packing the collection of objects into a fixed number of bins or containers. 1D bin-packing would have just variable lengths or widths; an example is a game of Tetris with just rectangles, except rotation is not allowed. 2D bin-packing would be the case of variable widths and lengths, a common example is minimizing material waste as seen in the clothing and manufacturing industries when dealing with fabric and sheet metal stock. Lastly 3D bin-packing deals with lengths, widths, and heights with fixed orientation and container constraints; this is a heavily researched problem in the shipping industry to ensure every truck or cargo plane is efficiently packed to maximize profit margins. Real world applications are rarely this simple, for example the shipping industry has to consider the weights of the boxes and yield priority to certain boxes that have time constraints and must be placed in a specific container. Figure 3.1 visually shows bin-packing applications.

Adapting the algorithms from 1D and 2D Bin-packing are often the best solutions to more complex bin-packing scenarios. The three best known algorithms in bin-packing are Next Fit, First Fit, and Ordered First Fit which were developed by Corcoran and Wainwright [21]. First Fit takes the items in the order that they come and places the items

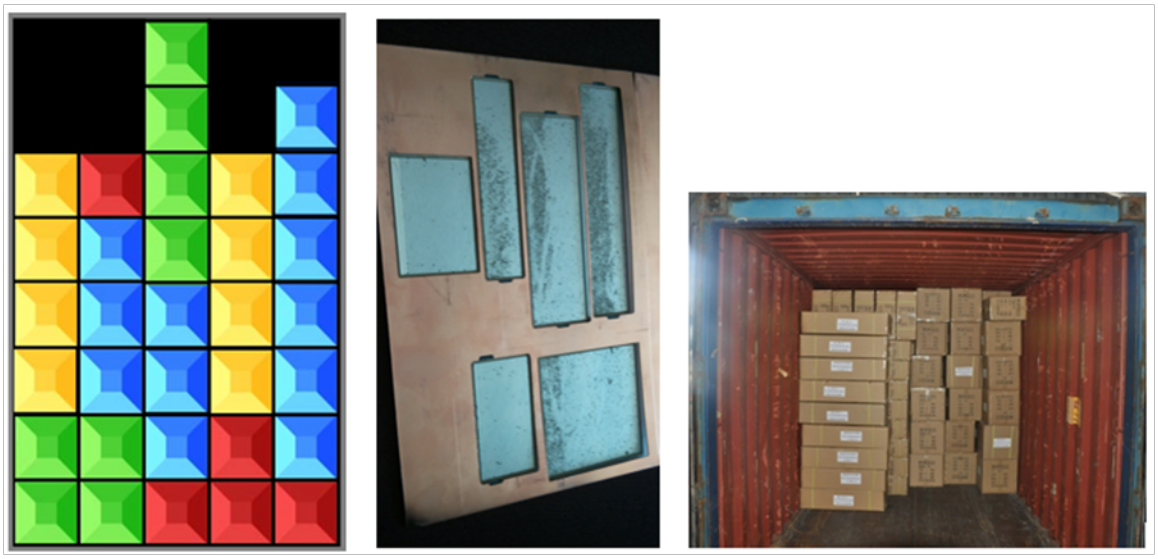

Figure 3.1. 1D, 2D, 3D Bin-Packing Examples

in the containers trying to minimize wasted space, very similar to the game Tetris. Next Fit is used in cases where there are multiple bins, or the original bin is split into slices. Items are places in the order they come in, but when an item cannot be placed in a bin it goes to the next bin. This helps pack large items that would not fit into nearly full bins. Ordered First Fit arranges the items from largest to smallest before placement and places the largest items first. This approach is seen in efficient suit case packing, as pants are packed before socks.

Although an optimal algorithm might not exist for a given bin-packing problem, mathematicians have proven the effectiveness of the common bin-packing algorithms. In 1973, Jeffrey Ullman at Princeton University proved that First Fit can be sub-optimal by as much as 70% [22]. In the early 1970's Ronald Graham and David Johnson of Bell Labs proved that the Ordered First Fit algorithm is never off by more than 22% [22].

Multidimensional bin-packing problems are rarely solved with an optimal algorithm as they are grouped into a branch of mathematics known as complexity theory [22]. A famous computational mathematical problem that has yet to be solved is the traveling salesman. This problem deals with finding the shortest route along a network of cities and roads, so that the salesman visits each city. It is easy to iterate through all possibilities in the simple case of 3 or 4 cities, but scaling the problem introduces endless possibilities that require exhaustive search.

Bin-packing in just 2 dimensions is subject to the same issues, as seen in the floor tiling problem that Erdos and Graham presented in 1975 [22]. The problem starts with tiling one million square inches of floor with 1 square inch tiles covering the maximum amount of floor. In this case, the optimal solution is simply a grid pattern with 1 trillion tiles and 100% coverage. Now consider the floor is $1,000,000.1$ in x $1,000,000.1$ in, the same solution would yield the same answer with 0.1 in strips of untiled floor on two edges. Erdos and Graham discovered a counterintuitive solution that was able to add 100,000 more tiles by skewing the tiles and introducing small gaps between them. To date their solution has not been improved upon nor has it been proven optimal [22].

3.2 Applying Bin-Packing in the VITAMIN Protocol

VCM in a dynamic link environment that requires predetermined mode selection must be done wisely to ensure working communications. All operating modes have different fixed symbol lengths. In the case of the VITAMIN system the 24 modes range from 22,080 to 571,456 symbols (assuming 16 frames per physical layer frame); at the common CubeSat symbol rate of 9600 S/s (symbols per second) that equates to 2.3 to 59.5 seconds which shouldn't be overlooked as the whole window of communications is 700 seconds or less. In 60 seconds the path loss can vary by 5 dB (depending on many factors), meaning the VCM protocol must consider the expected E_s/N_0 at all times and transmit or pack the PLFs in the right order and times to maximize the most information transmitted for the entire pass. This is a bin-packing problem.

The VITAMIN protocol is multidimensional in terms of bin-packing, as each mode operates above a certain E_s/N_0 threshold, pass time, and unique PLF length. Given the multiple variables no specific formula can be applied to a satellite pass that will guarantee the maximum data transfer. The approach to best solve this optimization problem is through a heuristic approach, mainly a highly flexible object orientated computer program. The program takes in all the various inputs needed to predetermine the coding and modulation for all time periods where communication with a ground station is possible.

Unlike the previous examples, where objects were being packed into physical space, VITAMIN is a little different. VITAMIN is packing information bits into packets that have an unique size (measured in time) and valued weight (information bits / symbol). These packets are created before transmission as needed to maximize throughput. The goal isn't to maximize packets placed, but to maximize the information bits transmitted. The

container is the pass, which has a width (time) and depth (SNR), only certain packets can go in certain zones as each packet has a minimum required SNR. Additionally there are infinite ordering possibilities for the packets, as time is continuous.

3.3 Program Overview

Several bin-packing algorithms are tested and measured for performance. An Ordered First Fit approach called First Choice and a slightly different Ordered First Fit approached called Top Down. Additionally Random Choice and Fixed Mode algorithms were investigated.

First Choice works from the start of the pass to the end. At the time to decide the transmission mode, a list of modes that meet the SNR threshold are determined. The mode that delivers the highest information per symbol will be used. Once the packet completes, the next packet mode is determined using the same approach. Since SNR can decline over time, packets that do not meet the minimum SNR available during the period of transmission are not selected.

The Top Down algorithm is similar to the First Choice algorithm, but places the first packet when SNR is maximum. Subsequent packets are then placed in time slots of decreasing SNR on both sides of the first placed packet until the entire time duration is filled. The motive for this algorithm came from the poor performance of First Choice in the cases where packets that deliver little information and take long amount of time to finish. Once the packet finishes the SNR climax could have come and gone, resulting in non-utilized path loss gain. The algorithms investigated and programmed are fully described in Appendix B. Additionally source code is available in Appendix A.

This thesis includes the development of the TMS (Throughput Mode Switching) program. It provides a solution to the bin-packing problem for the VITAMIN protocol. Additionally it is flexible enough to work with other VCM protocols or a subset of the 24 VITAMIN modes. TMS is also designed to interface with AGI's Satellite Tool Kit to provide monthly files for satellite uplink for any satellite and groundstation pair using a VCM communication system. The program runs on the Java Environment.

3.4 Determining the Shannon Hartley Limit

Determining the amount of throughput for each satellite pass using different transmission modes is only useful if there is a benchmark for comparison. Thus the Shannon-Hartley Theorem was applied to identify the theoretical maximum amount of data that can be

downlinked. This theoretical maximum is different than the theoretical VITAMIN maximum as the error correcting codes do not maximize capacity, this is seen in the separation between curves in Figure 1.3.

Shannon limit as described as $C = B\log_2(1 + SNR)$ (Equation 1.3) has two inputs, bandwidth and signal-to-noise ratio. Since the low Earth orbit passes have a time-varying link budget and are finite, the integral with respect to time is computed as

$$Throughput_{pass} = \int_{t_0}^{t} B\log_2(1 + SNR(t)) \text{ bits.} \tag{3.1}$$

Additionally the entire throughput for the entire data set is calculated,

$$Throughput_{total} = \sum Throughput_{pass} \text{ bits.} \tag{3.2}$$

3.5 Simulation Data Set

The VITAMIN protocol is intended to be flexible for all satellite orbits and ground station locations. For this reason the mode switching protocols need to be designed efficiently for all types of low Earth orbits. The TMS program can't simply look at one 15 minute satellite pass for one specific orbit and one specific ground antenna to determine the best algorithm for mode selection or to infer statistics of the VITAMIN protocol as a whole, because not every satellite pass is alike.

Since the major application of interest is CubeSats, four different types of Low Earth Polar orbits were analyzed. The four orbits considered are 600 km by 600 km at inclination angles of 70° and 98°, and 800 km by 800 km at inclination angles of 70° and 98°. Additionally the four orbits were analyzed for a high latitude location of Fairbanks, AK and low latitude location of Miami, FL. All eight communications scenarios were simulated via AGI Satellite Tool Kit for the month of October 2013 with 1 second data intervals.

All other variables of the link budget were held constant, thus only geometries of orbits and ground station locations play into the path loss calculation and length of communications.

3.6 VITAMIN Analysis from the TMS Program

3.6.1 Monthly Simulation

Table 3.1 and Table 3.2 show the monthly throughput for two ground stations located in Fairbanks, AK and Miami, FL. The throughput is expressed in GBits per month, which is assuming a 9600 S/s symbol rate and rather successful link budget. All 24 modes were available to the First Choice and Top Down methods. The Fixed mode column was set to only operate in mode 7 (Table 2.1), which might be typical of a basic CubeSat communication system. The assumptions dealing with the link budget include having a directional transmitted signal and a ground station with a large antenna gain (3 meter parabolic receive antenna). When comparing the two locations the throughput ratios are the more important statistic to analyze.

Table 3.1. Monthly Throughput (GBits) for Fairbanks, AK

Orbit (km)	Inclination	Shannon Hartley	First Choice	Top Down	Fixed Mode
600x600	70	6.65	5.09	5.13	1.76
600x600	98	7.31	5.51	5.55	2.11
800x800	70	7.15	5.38	5.44	2.15
800x800	98	8.11	6.03	6.09	2.63

Table 3.2. Monthly Throughput (GBits) for Miami, FL

Orbit (km)	Inclination	Shannon Hartley	First Choice	Top Down	Fixed Mode
600x600	70	3.11	2.36	2.38	0.881
600x600	98	2.88	2.18	2.20	0.819
800x800	70	3.56	2.66	2.68	1.15
800x800	98	3.30	2.46	2.48	1.07

All situations deliver approximately 75-78% of the theoretical maximum for the same link budget. Running the simulations at a symbol rate of 9600 kS/s (1000 times faster) minimizes the length effect of PLFs. These simulations improved by no more than 2%. A fully optimized VITAMIN protocol is estimated to do no better than 85% of the theoretical maximum (Shannon Limit). Any additional improvements would require different FECs.

In all eight cases Top Down outperforms the First Choice solution. The Top Down algorithm is used in all mode selection for the VITAMIN protocol. One other very slight advantage of the Top Down method over the First Choice method is that the transmission

timing is exactly centered along the pass. For example, assume a fixed mode case where a 6 minute packet must be transmitted in a 10 minute pass. Centered timing would result in the communication happening during minutes 2-8. The First Choice method starts immediately when communication is possible, meaning the communication would take place at minutes 1-7. This results in the Top Down solution operating above threshold levels slightly more than the First Choice solution, increasing the probability of success.

Two observations are made by repeating the experiment with the exact same ground station now located in Miami, FL. The VITAMIN protocol yields a similar 75% of the theoretical maximum throughput, meaning VITAMIN works to the same effect in all locations. The other interesting finding is the location of a ground station; having locations that can see the satellite as often as possible is important, especially if there is only one downlink. A design for traditional fixed mode system in Miami could be improved to double throughput by either relocating to Fairbanks or switching to VITAMIN, but why not do both to quadruple throughput?

3.6.2 Varying the Consecutive Number of Frames

Additional VITAMIN variables can be analyzed through the TMS program besides the timing of the mode switching. For instance each physical layer frame was designed to consists of 16 encoded frames and a 256 bit header [12]. Having fewer frames allows for overall shorter PLFs. Having shorter PLFs allows for quicker adaption to the link budget and the ability to switch modes quicker and thus increase throughput. Consequently, the more often a PLF ends the more frame markers are needed, thus increasing overhead. Figure 3.2 shows the performance differences. The same data set (section 3.5) was processed 17 different times using a modified VITAMIN protocol that consisted of a different Physical Layer Frame lengths. Additionally the three link budget scenarios were simulated to look at performance in different conditions.

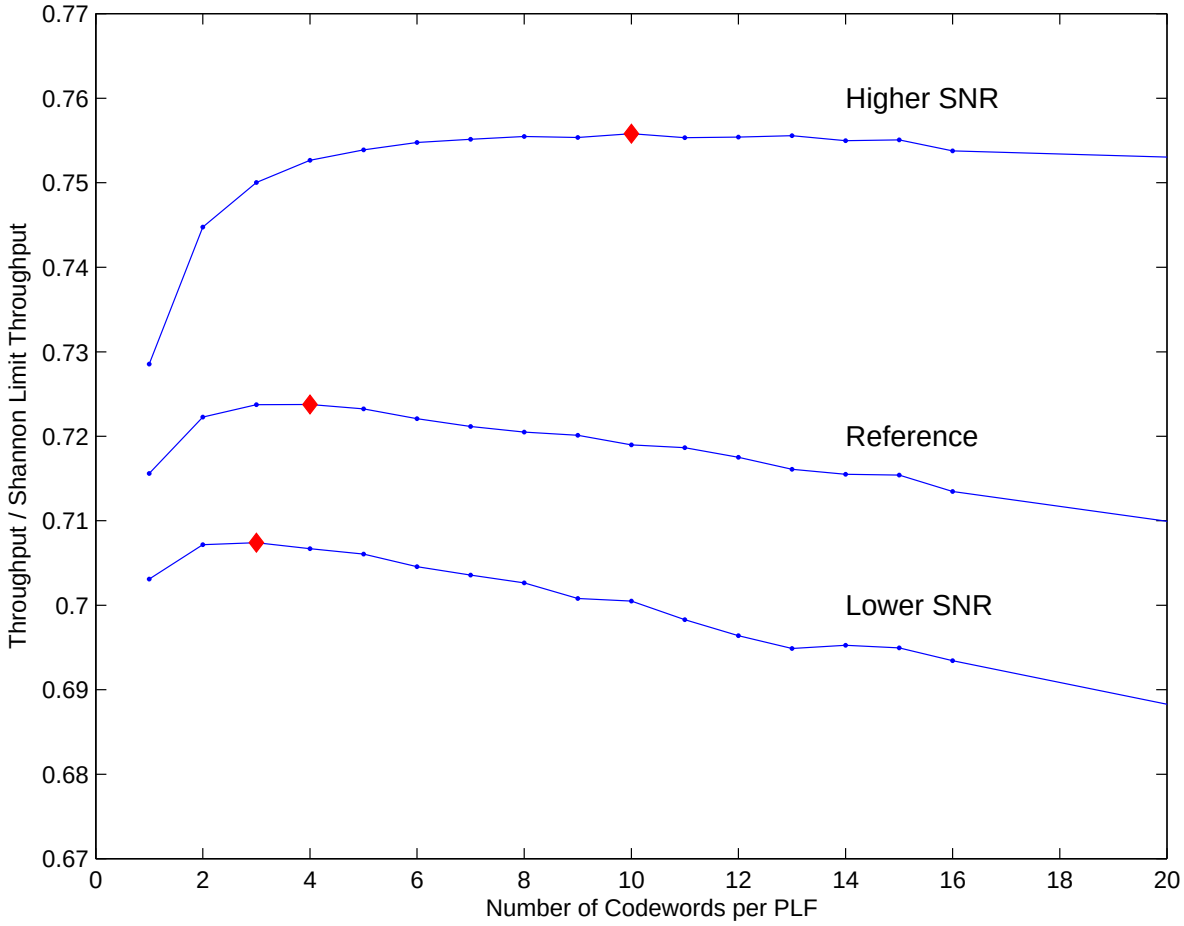

Figure 3.2. Varying the Consecutive Number of Frames in VITAMIN

Figure 3.2 showed three mainly flat curves, but all reaching a maximum in the 1-16 range. The lowest performing curve is a poor link budget, operating with modes in the lowest modulation orders. The highest performing curve is a good link, operating with modes in the highest modulation orders. The middle curve shows operation in an average link, in between poor and good. The maximum performances (red points) were at frame lengths of 3, 4, and 10 (poor to high). All simulations in all cases were within 4% difference for performance.

This analysis shows that throughput is only marginally affected by PLF length. Chapter 4 talks about implementation issues and makes a final decision on the number of frames to be used.

3.6.3 Identifying the Workhorses of VITAMIN

A commonly asked question of a VCM communications system is what operating modes are utilized most? This can be useful in the hardware design stage to simplify the communications system by excluding modes or modulations rarely needed to simplify circuitry and complexity. It is also interesting to the information theory engineer, as the dominant workhorse modes across the industry are further researched for their unique properties that make them more desirable. The TMS program calculates statistics and can list the modes transmitted and the percent usage. Table 3.3 shows the percentage used for the same complete data set discussed in section 3.5.

Table 3.3. Identifying Workhorse Codes

Mode Name	Percentage Used	Comment
LDPC 16APSK 1/2	14.6	Workhorse
Turbo QPSK 1/3	11.9	Workhorse
LDPC 16APSK 2/3	10.2	Workhorse
LDPC QPSK 1/2	8.54	Workhorse
LDPC QPSK 4/5	7.89	
LDPC 16APSK 7/8	6.08	
Turbo QPSK 1/4	5.70	
LDPC QPSK 2/3	5.42	
LDPC 32APSK 2/3	4.47	
Turbo QPSK 1/6	4.27	
LDPC 8PSK 1/2	3.92	
LDPC 32APSK 4/5	3.64	
LDPC 32APSK 7/8	3.00	
LDPC 64APSK 7/8	2.30	
LDPC 64APSK 4/5	1.89	
LDPC 16APSK 4/5	1.83	
LDPC QPSK 7/8	1.37	
Turbo BPSK 1/4	1.04	
LDPC 64APSK 1/2	0.940	
Turbo BPSK 1/6	0.511	Rarely Used
Turbo QPSK 1/2	0.415	Rarely Used
LDPC 8PSK 4/5	0.0613	Rarely Used
LDPC 8PSK 7/8	0	*Dominated by Higher Modulation
LDPC 32APSK 1/2	0	*Dominated by Higher Modulation

Since the 24 VCM modes cover a larger margin of E_s/N_0 thresholds than one ground station - satellite pair would experience over time, the workhorse simulation is an average of 16 individual workhorse simulations covering all ranges of the link budget to ensure the

lower threshold and higher threshold modes get the same opportunities.

Of the 24 modes, 2 modes LDPC 8PSK 7/8 and LDPC 32APSK 1/2 never get a chance as other modes can deliver more information at or below their minimum thresholds. Equal distribution would equate to 1/22 or 4.5% percentage usage. 4 Modes fall into the 75^{th} percentile which puts them in the workhorse category, LDPC 16APSK 1/2, Turbo QPSK 1/3, LDPC 16APSK 2/3, LDPC QPSK 1/2. Three modes fall are in the 25^{th} percentile and are rarely used, TurboBPSK 1/6, Turbo QPSK 1/2, LDPC 8PSK 4/5.

3.7 Conclusions

The Top-Down algorithm was ultimately decided on as an empirically good solution, but not until it was apparent that it outperformed the other algorithms on a large data set. Having the best method for implementing a VCM protocol is just as important as having a VCM protocol in the first place. While certain algorithms might only outperform others by a sight margin for every satellite pass, over the course of time the difference adds up. The additional calculations in software improve the overall system and are worth the extra time in design.

There is evidence to redefine a Physical Layer Frame as a Frame Marker and 1 codeword, instead of 16 codewords. This is supported by Figure 3.2. Additionally, the hardware implementation of Chapter 4 becomes simpler as FMs are inserted more often. At the same time VITAMIN realizes that the number of codewords per PLF might be optimal at different values depending on the application itself, and conveniently in VITAMIN the modification of this parameter will not have ripple effects.

Chapter 4
VITAMIN Radio Implementation

4.1 Introduction

The most valuable way to investigate, measure, and analyze a communication protocol is to build it and try it. This chapter does exactly that with the VITAMIN protocol, as the objective of Chapter 4 is building a transmitter and receiver that use VCM. Instead of designing hardware specifically for CubeSats, flexible existing hardware was chosen to keep the focus on the VITAMIN protocol. The radio implementation in this chapter was done solely with two software defined radios, each paired with standard computers. This chapter goes over the development process, displays the final end-to-end communications system, addresses new findings, and grades the performance of the VITAMIN protocol.

Unlike the MATLAB simulation in Chapter 2, this radio implementation puts the data into the air (or a coaxial cable) with an S-band carrier frequency (2.2 GHz) and then receives the signal with independent hardware. This chapter goes through building and operation of both the transmitter and receiver. Additionally, a fully functional software defined radio receiver paired with a computer is a perfect and simple ground station receiver. In the case of globally distributed ground stations, off the shelf hardware and software is the easiest way to grow a reliable ground station network. For those reasons, the majority of this chapter is about the receiver.

Only BPSK, QPSK, and 8PSK modulations were implementing in LabVIEW. For the first attempt at a software defined receiver, not implementing Amplitude Phase Shift Keying keeps everything simpler. Using only modes 0-15 in VITAMIN is reasonable because they still cover many information rates and different operating thresholds. For this reason, mode 13 is implemented in LabVIEW as mode 16 is not available.

4.2 The Hardware and Software

The radio implementation uses two National Instruments Universal Software Radio Peripherals (USRPs) model NI-2920, pictured in Figure 4.1. The core of these radios are the analog-to-digital converters. To support a wide variety of radio frequencies, interchangeable daughter boards are switched as needed. The WXB daughter board was the main board used in development as it supports both transmit and receive in the from 50 MHz to 2.2 GHz. For support of signals at an intermediate frequencies the Ettus LFRX and LFTX daughter boards were used, capable of 0-30 MHz. The USRPs have SMA connectors

Figure 4.1. Lab Setup - 2 NI USRP 2920

for connecting to antennas or directly to each other through a coaxial cable and optional antennuators.

From the USRPs the digital representation of the analog baseband waveform is streamed to an interfacing computer via Gigabit Ethernet. The USRP is supported by many software clients including MATLAB, GNURadio, Python executables, and LabVIEW. LabVIEW was the software platform selected after initial investigation of all platforms. Reasons for this selection included more detailed documentation and a larger selection of included library functions. GNURadio appeared to be a viable alternative and should be able to accomplish the same functionality.

Communication between the USRPs and the computers is handeled by LabVIEW drivers. Initial user input includes the TX/RX frequency, IQ sample rate, IP address of the USRP, and the TX/RX gain. Thereafter LabVIEW can continuously fetch or transmit samples and then close the connection. The USRPs also have the ability to share information and communicate with each other through a MIMO (multiple input multiple output) cable. This was used solely for debugging parameters like internal vs actual frequency offset. Figure 4.2 shows the high level flow of data, from LabVIEW on a computer to a USRP, into the air, received on another USRP, and recovered in LabVIEW on another computer. All arrows are bidirectional, because the hardware supports duplex communications, but

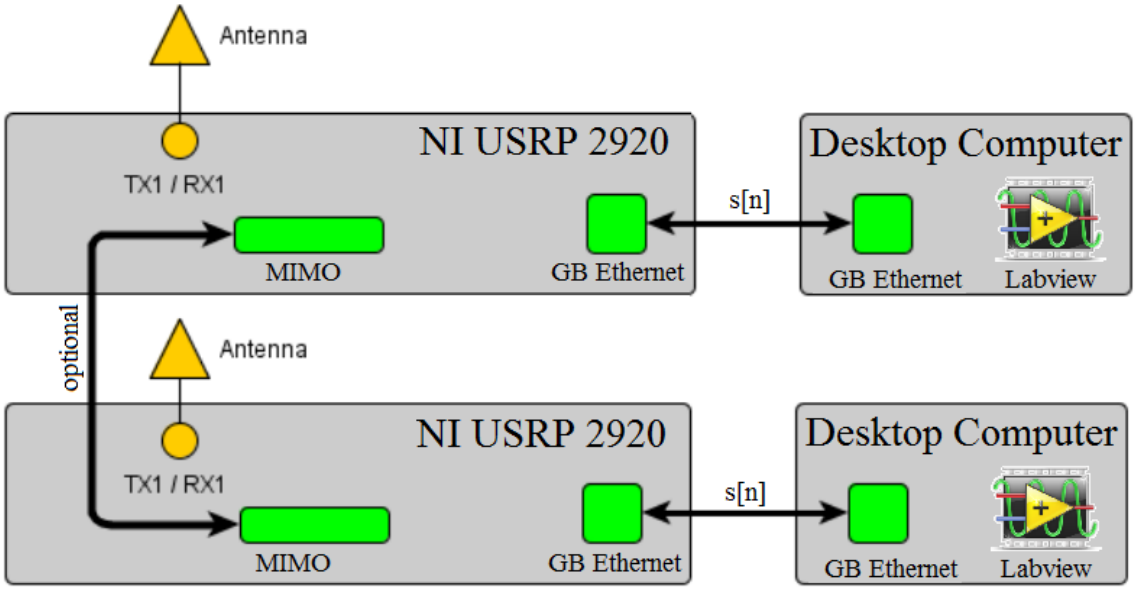

Figure 4.2. Software Radio and Computer Connections

in this experiment data only flowed in one direction.

4.3 Data Flow Overview

Figure 4.3 shows how the information bits move through the software and hardware that implements the VITAMIN protocol. More detailed figures of both the transmitter and receiver are shown in Figure 4.4 and Figure 4.8, respectively.

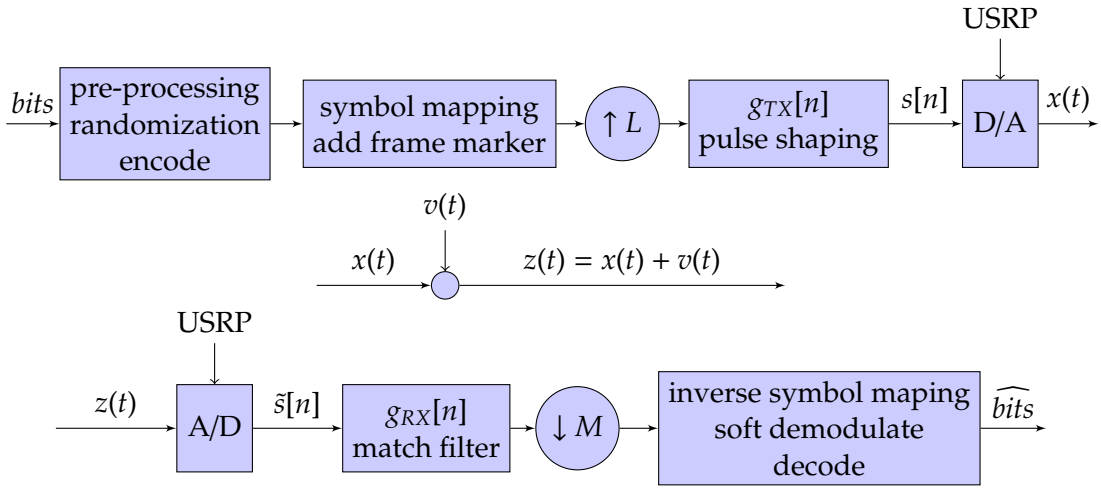

Figure 4.3. Transmitter, AWGN Channel, Receiver

- $s[n]$ represents the digital complex waveform that is exchanged between LabVIEW and the USRP, $s[n]$ is also labeled on Figure 4.2.
- $x(t)$ is the analog signal on the carrier frequency.
- $v(t)$ represents any noise introduced to the signal.
- $z(t)$ is the signal with noise that enters the receiving USRP
- $\uparrow L$ and $\downarrow M$ represent the upsampling and downsampling, which is necessary to accurately represent the analog waveform on the USRP by having multiple discrete data points for every symbol.

4.4 Transmitter Development

Using a USRP and desktop computer, a VITAMIN transmitter was developed. Although future work for VITAMIN in space will involve a CubeSat sized transmitter, having a USRP transmitter allows for testing of both the receiver and the overall protocol. The flow diagram in Figure 4.4 shows the transmitter processing stages. Initially one time processing is done for the matched filter coefficients, generation of the complex frame marker array, radio parameter setup, and the user variables listed in subsection B.4.1. After that point the transmitter continuously transmits Physical Layer Frames based on the mode number specified by the user.

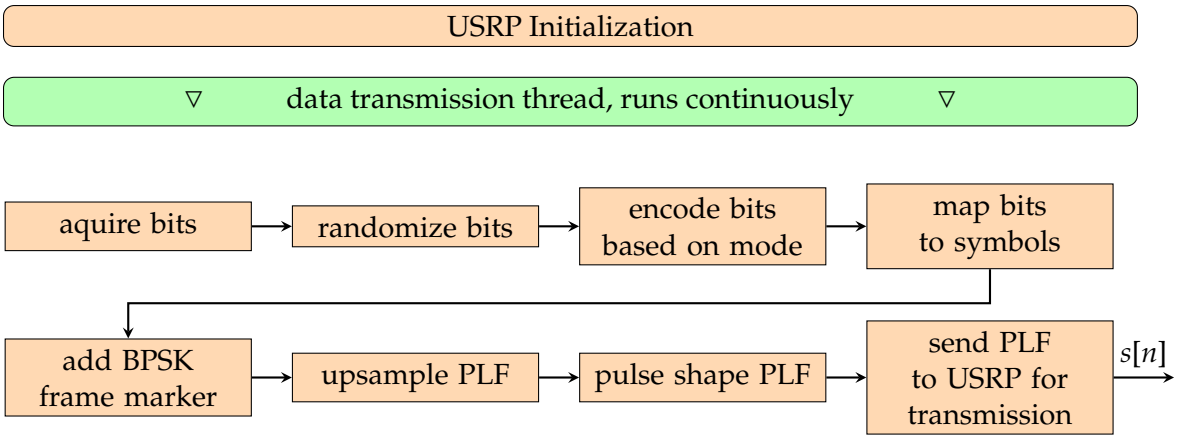

Figure 4.4. Transmitter Flow Diagram

4.4.1 Transmitter Initialization

Before the first symbol is transmitted, the transmitter has to gather the user input and execute a few tasks. Figure 4.5 shows the graphical user interface of the transmitter. White boxes are user inputs and gray boxes display operational values. All user inputs and output are explained in detail in subsection B.4.1. The transmitter will repeatedly run a mode if a valid mode number is specified or if mode 99 is selected it will read an input file that consecutively lists what modes to run in order. Symbol rate is always determined by IQ sampling rate divided by samples per symbol. Optional Additive White Gaussian Noise is implemented before transmission, and can be switched on by a push button.

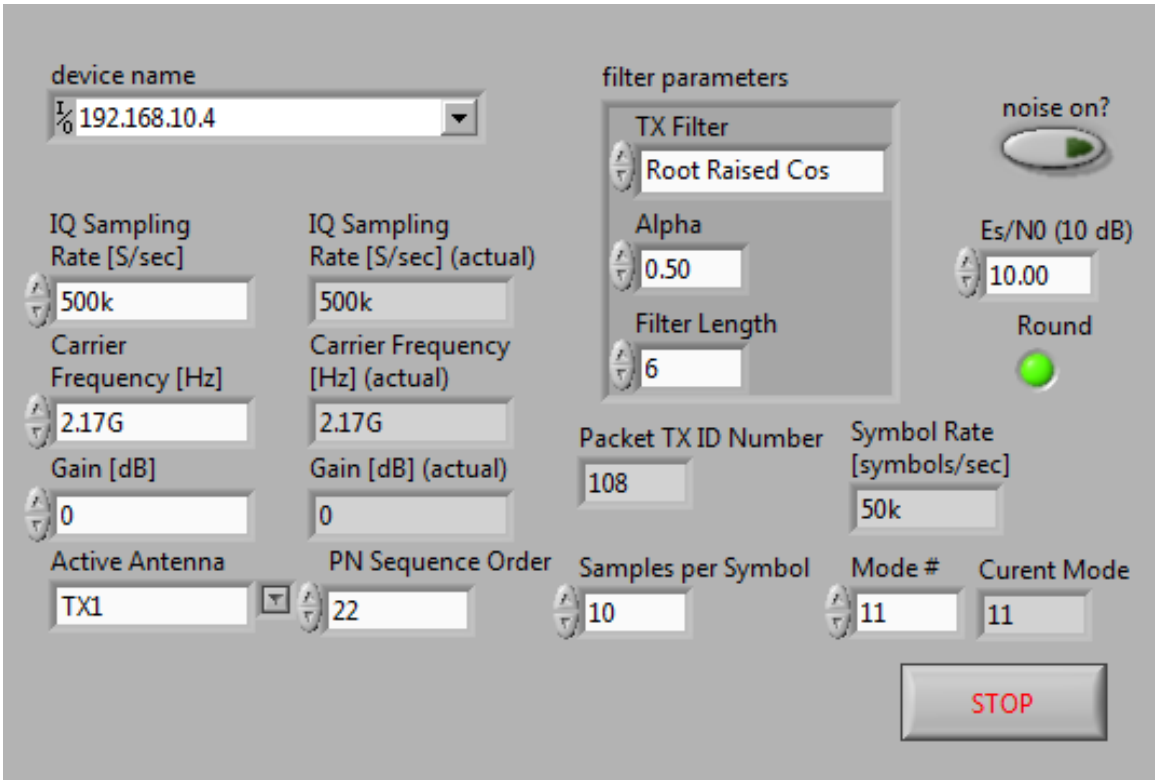

Figure 4.5. LabVIEW Transmitter User GUI

Behind the scenes, the transmitter needs to initialize arrays for encoders, the static frame marker, and pseudo random information bits. Communication needs to be established between the computer and USRP, and the parameters of carrier frequency, gain, antenna, and IQ rate need to be initialized. After this point the main data transmission thread takes over, continuously transmitting PLFs.

4.4.2 Data Transmission Flow

The first step to building a PLF is slicing the information bits. This is because the modes selected in VITAMIN protocol only have three information frame lengths. These lengths are 7136, 8920, and 16384. Since the program is designed to use a pseudo random sequence the correct length of bits is generated. The next step is randomization of the information bits to create an equal number of 0's and 1's and evenly distribute them. This pseudo randomization is consistent with [12] and explained in subsection 6.3.2.

At this point the information bits are encoded to transmission bits that contain the forward error correcting properties. Next, the bit stream is mapped to a complex waveform of symbols determined by modulation type. Two modes have encoded lengths that are not divisible by their modulation order M; modes 12 and 14 both have a remaining 2 bits to map where the symbol consists of 3 bits, thus an extra zero is appended.

The BPSK modulated (symbol mapped) frame marker is attached to the encoded symbols creating a complex waveform that is in Physical Layer Frame form. The PLF is upsampled and put through a pulse shaping filter. The PLF in the complex waveform, $s[n]$, is sent to the USRP via Ethernet to be transmitted at the carrier frequency. At this point the cycle continues with the next PLF. To prevent the underflow of data in the transmission process LabVIEW shift registers are used between lengthy operations to ensure the next frame is always ready and the USRP isn't waiting on the encoder. For example, whenever a frame is transmitted on a USRP the next frame is being pulse shaped and the frame after that is being encoded.

4.5 Communications Channel

Using the USRP there are several options for transmission mediums, carrier frequencies, and noise sources. Two UHF antennas were used for over the air transmission with the WBX daughter-boards, rightmost PCB board in Figure 4.6. The other option is communication via coaxial cable, which doesn't require specific antennas and is the best method for low noise communications.

Transmission over the air introduces multipath and signal degradation from non-directional antennas. Signal attenuation can be used to reduce the signal strength of the source to levels where the receiver sees the effects of thermal noise. Additionally AWGN can be added digitally at the transmitter to produce a signal with a range of E_s/N_0 levels. Figure 4.7 shows the what the transmitted signal looks like on a spectrum analyzer at 2.2

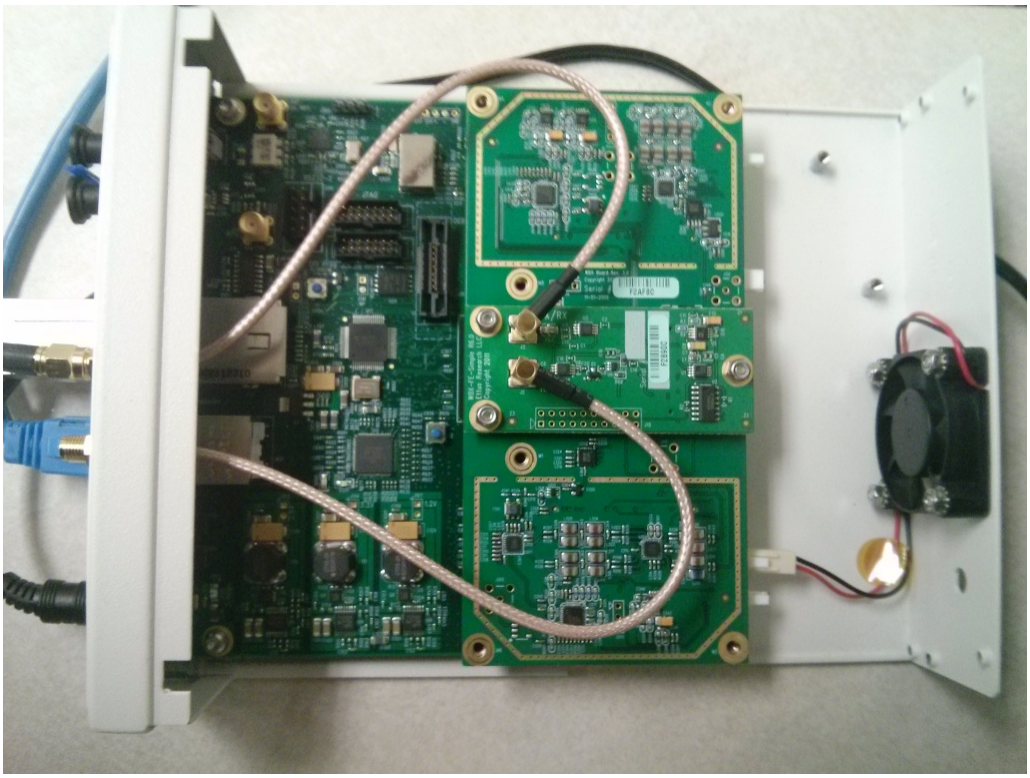

Figure 4.6. Inside of USRP 2920 with WBX TX/RX Board

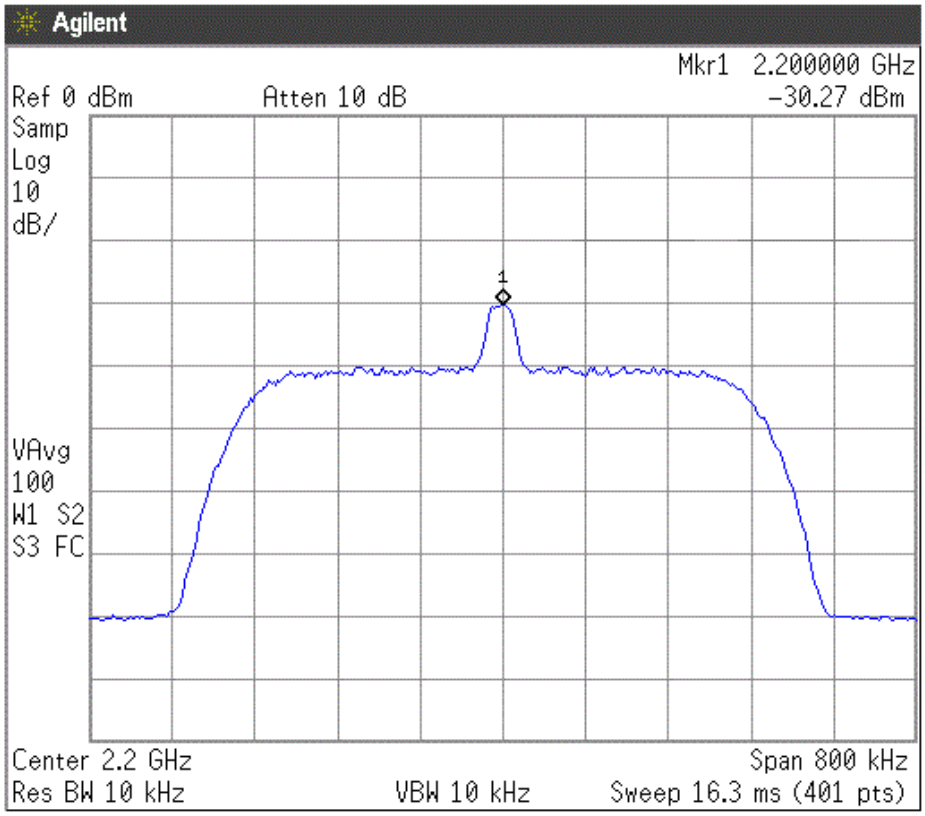

Figure 4.7. Spectrum of USRP Output for Mode 11 at 50 kS/s

GHz.

4.6 Receiver Development

Figure 4.8 shows the basic flow of data in the receiver from the moment the radio signal is captured till the point the information bits are recovered. The LabVIEW receiver, is designed to be either on or off. Once it is told to run the first step is initializing memory, then two core processes take over: data collection and data decoding.

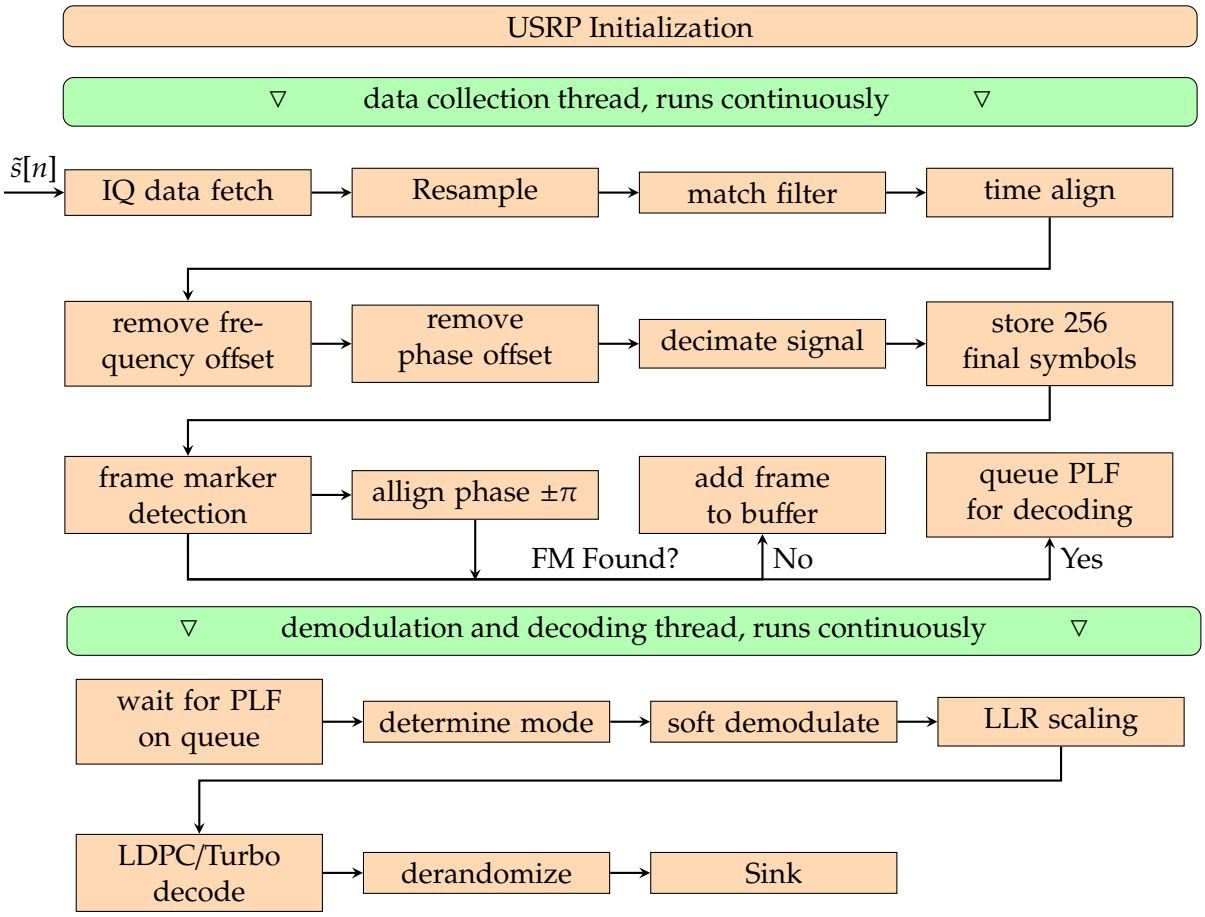

Figure 4.8. Receiver Flow Diagram

4.6.1 USRP Initialization

The USRP needs several parameters to be set correctly to successfully operate as a receiver. The IQ sampling rate is bounded by the minimum and maximum supported by the hardware of 195.3125 to 2,000 kilo-samples per second. The oversampling factor (number of samples per symbol) is used for determining the ideal sampling point of the incoming waveform. Matched filter parameters are identical to pulse shaping parameters of the transmitter. Acquisition duration, measured in seconds, tells the USRP how long to fetch samples. When activated, a switch called expecting packet number, tells the receiver to extract the packet ID numbers. The reference frequency source, allows the receiver to use an internal carrier reference frequency (default) or the transmitter's frequency via the MIMO cable. Other inputs include IP address, PN sequence seed, receiver gain, and antenna

selection. Figure 4.9 shows the LabVIEW receiver GUI.

After the receiver is initialized with the input parameters, memory is allocated for the various arrays and buffers. The expected random data is pre-generated. Before the fetching and processing state of the receiver becomes active, one initial Frame Marker must be detected first to lock the re-sampler, demodulator, matched filter, and decimator to their steady states.

4.6.2 IQ Data Fetch

The incoming waveform needs to be continuously collected and processed. In theory the minimum number of symbols needed to be collected at one time is 256, which is the frame marker length. Since the receiver is designed to detect at most one frame marker per frame, the VITAMIN protocol's shortest mode length determines the maximum fetch size. Since the next frame of data is being buffered while the previous frame is being processed, both underflow and overflow conditions must be avoided. For example a large fetch size can result from high upsampling factors that unnecessarily increase memory allocation and slow down subsequent functions in the data flow. Likewise too small of a fetch will shorten the allotted time for data processing. The acquisition duration and in-phase and quadrature data (IQ) sampling rate product directly determine the number of IQ pairs fetched.

4.6.3 Resample

The IQ sampling rate divided by the upsampling factor yields the maximum symbol rate, but since this might be higher than the actual symbol rate the receiver has a specific symbol rate input. A resampler function will resample to the desired lower symbol rate. The signal is then at the correct symbol rate, but is still oversampled (contains more data points than used in decoding).

4.6.4 Matched Filter, Time Align, Frequency Offset, Phase Offset, and Decimation

Once the fetched frame of complex IQ pairs are acquired and resampled, a few operations are required to be performed to the incoming signal before the frame marker is detected. Matched Filtering, Frequency Offset Correction, and Phase Offset Correction are all handled by one subsystem that takes in the complex waveform, expected symbol mappings,

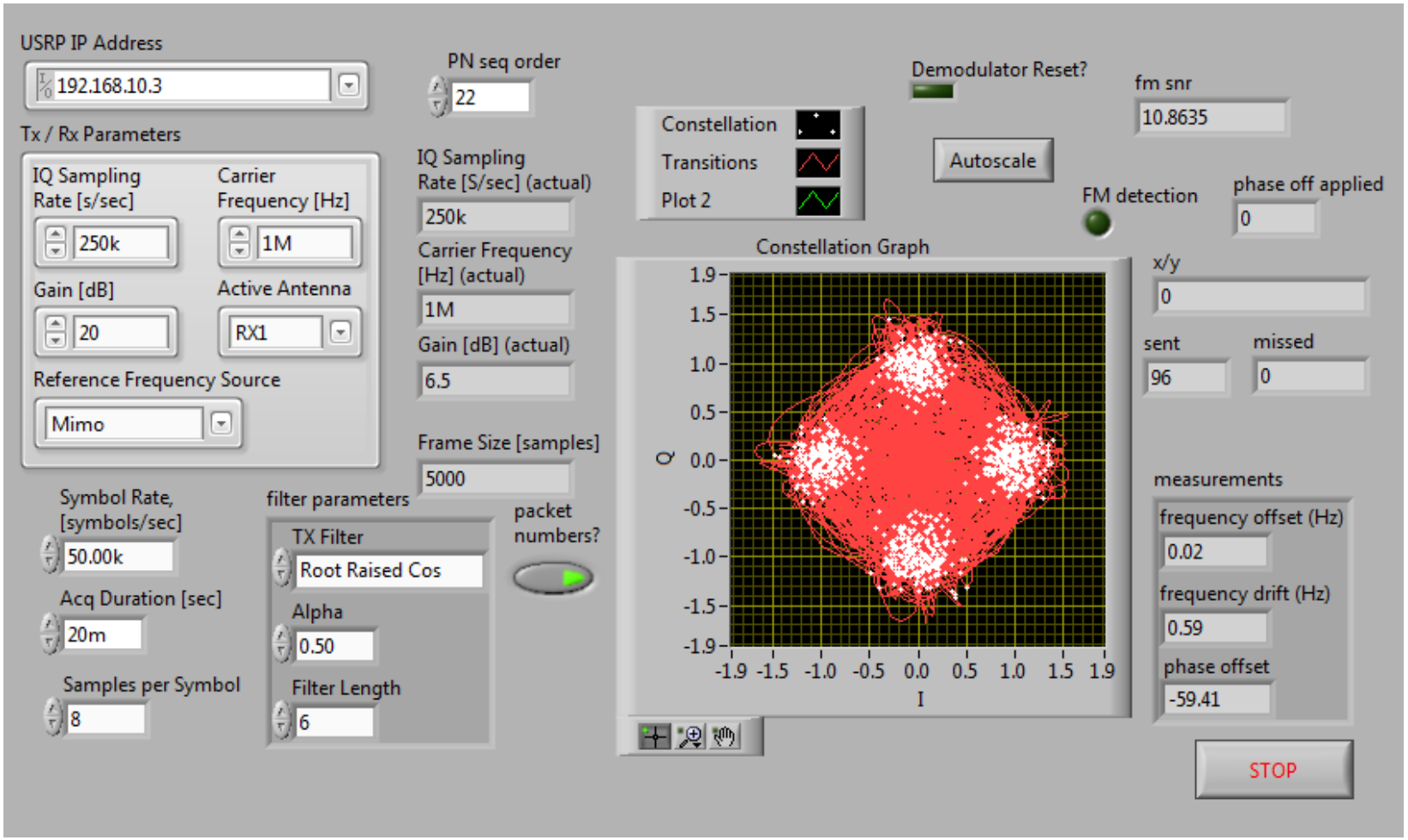

Figure 4.9. LabVIEW Receiver User GUI

and frame marker sequence. An existing LabVIEW resource called "MT Demodulate PSK" was used to perform all three of these operations. The output taken is only a modified complex waveform. The signal is first put through a matched filter to remove any inter-symbol interference. A max eye technique is used to determine the ideal sampling point, meaning whichever sampling point that produces the least amount of inter-symbol interference is chosen. The ideal sample point is then shifted, which prepares the signal for easy decimation. Next the frequency offset is accounted for and corrected. The last two modifications are gain scaling and any phase offset correction. Decimation is immediately performed through another subsystem. Since the output signal is aligned the decimator simply takes every x^{th} bit where x is the upsampling factor.

The MIMO cable was used to remove frequency and phase offset effects from the receiving system for testing purposes. This cable allows the transmitter to share its exact carrier frequency and clock with the receiver as these are the best references available. Results show that frequency offset when corrected internally provides the same performance in bit errors as the MIMO reference. The USRP did show significant issues with recovering the correct phase offset under very noisy conditions, even with the MIMO reference. This is discussed further in section 5.5.

4.6.5 Frame Marker Detection

The frame marker's short 256 bits are extremely important in the receiver's operation. Not only does it announce that a physical layer is beginning and data frames are following, but the detection of two frame markers is the VITAMIN mode identification technique as Chapter 2 proposed. The URSP is configured to fetch data in packets with sampled lengths ($fetchedlength = upsamplingfactor \cdot samplefetchsize$) that must be smaller in length than that of VITAMIN's shortest mode PLF length. This means, in any given data fetch, the fetch may or may not contain a frame marker, but it definitely will not have two FMs. Two FMs in one fetch is uncommon since most of the up sampled symbol lengths are very long, but in the case of mode 27 (64APSK) the frame is only 1360 symbols. To keep implementation simple, and not adding the extra complexity of searching for two frame markers, the minimum number of symbols to process in the LabVIEW receiver is no more than 1360. Additionally there is the possibility that a data fetch can contain a partial FM, and that case is accounted for by sharing information across iterations.

The frame marker correlator function takes in four arguments. The first is the newest

filtered and corrected fetched IQ data, the seconded is the previous 256 symbols of IQ data, the third input is previous phase correction, and the last input is the correlation threshold. The previous 256 symbols are need in case the first 255 bits of the fetched data contain the 256 bit frame marker. The frame marker is found by performing a cross correlation on the concatenation of the 256 bit previous fetch with the current data fetch and the known BPSK frame marker.

For receiver processing reasons, this implementation treats BPSK as a QPSK subset that only consists of (0+1i) and (0-1i), see section 4.8 for more details on this. Knowing this the demodulator can place the frame marker in two positions, on the I or Q axis. The sliding cross product is processed across all 256+N symbols and is tested at each step to determine if a threshold is met. A perfect cross correlation value for a 256 symbol frame marker has a magnitude of 256, this is plotted in Figure 4.10. In Chapter 2 it was discovered that false positives only occur at E_s/N_0 levels of -8 dB when the threshold is set at magnitude 200, hence the detector uses this threshold as default. Every simulation and interaction with the LabVIEW system confirmed this finding; frame marker identification was performed perfectly for E_s/N_0 ratios above -8 dB.

Initially simple LabVIEW dot product functions and "for loops" were used to compute the cross correlation. Although this worked, it was a slow technique and created a processing bottleneck. Knowing this, the CrossCorrelation.vi was used in conjunction with the ThresholdDetector.vi to accomplish the same thing at a much faster speed.

The frame marker is used to apply the correct phase alignment of the incoming signal for BPSK, QPSK, and 8PSK. As long as the frame marker is aligned in the Q or imaginary axis the following complex symbols regardless of modulation will be aligned because the transmitter is designed not to introduce any unnecessary phase shifts when switching modulation types. Once the correlator finds the frame marker, it determines the phase offset and corrects for it. If the frame marker is not found, the best the receiver can do is assume the same phase offset that was previously received still holds true and apply that. For most cases this works just fine, but in cases where there is antenna movement or noisy conditions phase ambiguity will lead to errors.

Lastly the frame marker detector will return the index where the data frame starts. In the case that the frame marker isn't found, a -1 is returned. In the special case that the frame marker is the last 256 bits of the received frame, the correlator will not report it, but let the next iteration find it, as it will also exist in the 256 bit remainder. This simplifies things,

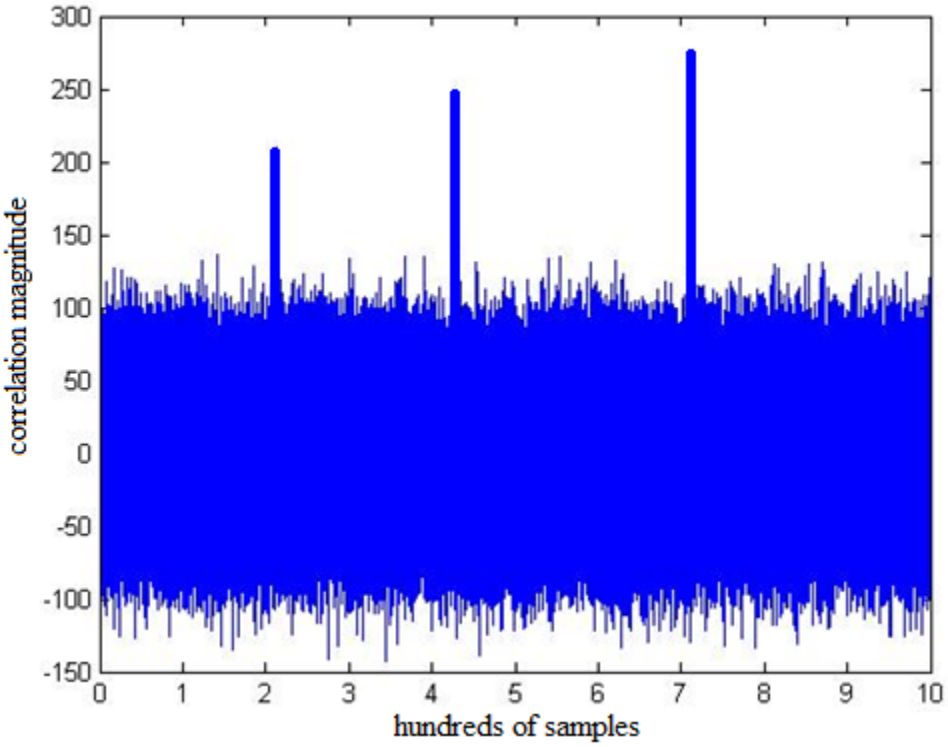

Figure 4.10. Detectable FM Correlation Spikes with E_s/N_0 -8 dB

so the correlator never has to report that first information symbol starts in the previous or next packet of data.

4.6.6 Aligning Phase

Before the incoming IQ waveform is soft demodulated and decoded, the phase offset needs to be applied. This happens right after the frame marker correlation process which determines the offset value. In the case of the LabVIEW implementation the offset can be any multiple of $\pi/4$. The previous phase offset value is kept in memory throughout the data fetch operations. The majority of the fetches collect only information data and any offset that was applied previously will have to be carried out until the frame marker detector can update the offset again. The demodulator attempts to correctly track any phase jitters so that abrupt phase flips should only occur when signal is interrupted.

4.6.7 Queuing of PLF

Once the second Frame Marker is found, a physical layer frame becomes complete. The resulting frame of symbols is counted for its length. If it is an unexpected length the frame is immediately discarded, otherwise the frame is put onto a FIFO queue system and the data collection thread ends here. The demodulation and decoding threads start up once the queue contains frames.

4.6.8 Determining the Mode of PLF

The LabVIEW receiver is designed to have 1 or 2 threads processing incoming frames. This is mainly due the lengthy decoding time of the turbo demodulators. It is also possible to allocate more threads on more powerful computers to in increase the maximum iteration number. The threads wait until the queue contains a frame. Once the frame is gathered, the first task of the decoding thread is determining the mode. This is easily done by using the frame size and a lookup table.

4.6.9 Soft Demodulation

Once the frame is gathered, the array of the complex IQ pairs is sent to either a BPSK, QPSK, or 8PSK soft demodulator. Unlike a hard demodulator where bit decisions are made, soft demodulators provide the decoders with information about the uncertainty of the received symbol. The BPSK soft demodulator simply takes the complex array and selects the imaginary component and stores it into an floating point array. Similarly the QPSK soft demodulator converts the N symbols into 2N soft bits, using a identical process to BPSK, but in both the I and Q channels. The 8PSK demodulator does the same as it converts N symbols into 3N soft bits, but the process is not as trivial. Each complex symbol's distance is calculated for each of the three possible bits for all 8 constellation points, the shorter the distance the more likely the bit is a 1 or 0.

4.6.10 LLR Scaling

LLR stands for log likelihood ratio and allows the decoder to use the estimation of the noise in the signal to converge on a solution faster when SNR is high. Ideally the signal strength information would be provided by the radio for all time samples, but this is not the case with the USRP. The solution was to have all three demodulators estimate the noise

of the incoming signal by the variance of the IQ pairs with the expected symbol mappings. For example a noiseless demodulated BPSK signal would only consists of ±1, so finding the variance of the actual signal with the expected provides a good estimation of the SNR. Since all IQ pairs have an equal amplitude, the LLR scaling is $2y[i]/\sigma^2$.

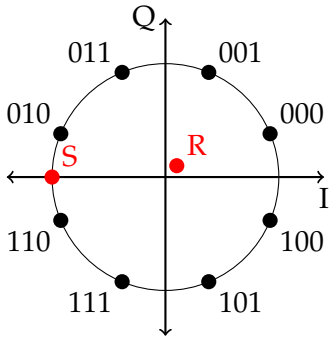

Figure 4.11. Two Complex Symbols

Figure 4.11 shows two symbols R and S. In the programming code an 8-bit output is used for all the soft decisions. The scale is -127 to 127, where positive numbers represent 0 and negative numbers represent 1. The larger the magnitude of the number the more certain the decision, for example, -10 represents a 1 of low certainty while +120 represents a zero of high certainty. Since R lands almost at the origin at $(0.1 + 0.1i)$ the soft decision is close to complete uncertainty resulting in soft bits of (38,38,0), being slightly northeast results in a higher probability of 0 for the first two bits. Point S at location $(-1 + 0i)$ results in (0,-127,100). Even though S falls directly in between two constellation points the last two bits are very likely to be 1 and 0, this is because of Gray coding. The first bit has compete uncertainty since it has no quadrature component; in this 8PSK constellation all 0's have a positive quadrature component and the 1's have negative quadrature. The interesting observance is the that the soft demodulator shows how there is more certainty in the middle bit. This is because the four closest constellation points all share the same middle bit, while the right most bit does not. Soft demodulation algorithms are from equation 16 in [18].

4.6.11 Decoding

Once the signal is demodulated the frame is put through one of the three decoders: turbo, AR4AJ LDPC, or C2 LDPC. The decoders expect the LLR input, which is discussed in the previous section.

All error correcting decoders are implemented in C code instead of LabVIEW code. These decoders were provided by the Jet Propulsion Laboratory. Code dealing with interfacing and memory management still need to be done before the C decoders can be run. To access the decoders in LabVIEW, the C decoders and dependencies were compiled into seperate DLL files that are callable from LabVIEW. The inputs are the information frame array, code rate, maximum iterations, and a preallocated array for the output bits.

The rate of the code is also passed onto the decoder as each coding rate has a specific puncture pattern which produces the right result for the right rate, as the functionally of the decoder is identical for every rate.

All the decoding types were independently tested outside of the VITAMIN implementation in LabVIEW and are functional with the expected performances. The LDPC decoders are much faster in processing time than the turbo codes.

4.6.12 Derandomization

Once the information bits are recovered they still need to be dereandomized. This pseudo randomization process is consistent with [12] and explained in subsection 6.3.2.

4.6.13 Sink

The receiver finally checks the bits with the expected bit stream and determines if the packet was successfully recovered. Additionally the receiver will recover the packet number that is embedded into the data payload. Using sequential packet numbers and recording which packets the receiver successfully decoded, is a simple and easy technique to determine the codeword error rate.

4.7 Pulse Shaping and Matched Filtering

A finite impulse response (FIR) filter is used in pulse shaping and matched filtering. Digital signal processing requires pulse shaping for maintaining a band limited signal. The pulse shaping introduces a controllable amount of inter-symbol interference that can be removed by the corresponding matched filter [23]. Equation 4.1 describes the digital signal filtering process in LabVIEW,

$$y[n] = b_0 x[n] + b_1 x[n-1] + ... + b_p x[n-P], \tag{4.1}$$

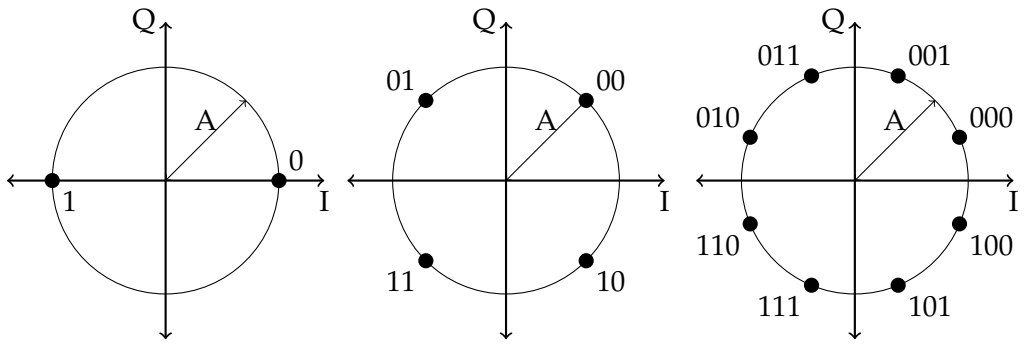

Figure 4.12. VITAMIN Signal Constellations

where P is the filter order, $x[n]$ is the input signal, $y[n]$ is the output signal, and b_i are the filter coefficients.

4.8 Constellation Workaround

LabVIEW support for mixing modulation types is limited. The Modulation Toolkit (MT) Demodulate PSK File-extension: Variable Information (VI) was used for recovering the complex waveform after matched filtering, symbol timing, and frequency/phase offset corrections were applied to the raw input signal. The VI is called "demodulate" and has the ability to output hard decisions, but this feature was not used. All decoders require soft demodulation. The MT Demodulate PSK VI uses a known symbol map input for its calculations and symbol mapping. Knowing this the first attempt at VCM demodulation was to split the signal and run BPSK, QPSK, and 8PSK demodulators in parallel with their respective symbol mappings. Once two frame markers determined the mode, the two unneeded waveforms were discarded from memory. The problem with this technique was keeping all three demodulators locked. VITAMIN defines the signal constellations as seen in Figure 4.12. The issue with this is the transitions from one modulation type can create a phase change of $\pi/8$ which isn't expected by any demodulator, plus lower level demodulators will not lock when higher order modulations are being received as there are unexpected phase shifts. Although the demodulators can quickly lock once the signal appears, the first part of the PLF could be lost, since VITAMIN doesn't provide any sync or guard bits.

Modifying and rebuilding the digital LabVIEW demodulators from scratch would be a time consuming task that requires detailed knowledge and experience in the DSP of the USRP. Thus the method of altering the constellations became the most practical solution.

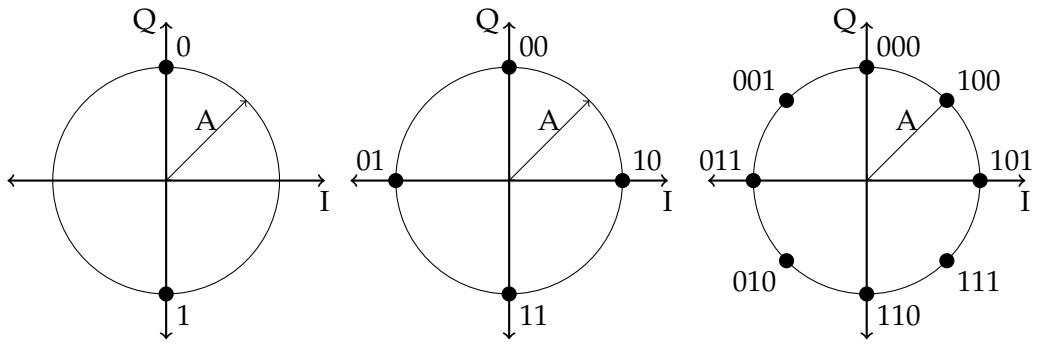

Figure 4.13. Signal Constellations Used for VITAMIN in LabVIEW

Figure 4.13 shows the implemented constellations.

Figure 4.13 is identical to Figure 4.12 , just rotated (QPSK by $\pi/4$ and 8PSK by $3\pi/8$). The rotation means the QPSK and 8PSK demodulators are unaffected with regards to locking and symbol syncing when a BPSK FM passes through them, as the FM will always land on a valid constellation point, thus removing the $\pi/8$ transitions. The new mapping allows for smoother transitions in the transmitter, as only phase shifts that are multiples of $\pi/4$ are produced when switching modulations.

One prevailing issue is keeping the lower order demodulators locked when a higher order mode is being transmitted. For example, when the PLF is 8PSK modulated, the QPSK demodulator will become unlocked as one half of the constellation points will fall in between valid QPSK constellation points. This was solved by having the LabVIEW receiver demodulate all PLFs with the 8PSK demodulator. This works because BPSK and QPSK can act as subsets of 8PSK so the IQ points will be correctly mapped. However this solution is not optimal, under noisy conditions as noisy symbols will be mapped to constellation points that are not valid for lower order modulations. It is important to note that codeword error rate simulations are done with the exact symbol mappings corresponding to the modulation type expected to remove the possibility of demodulation (symbol mapping) errors.

Chapter 5
VITAMIN Performance

5.1 Codeword Error Rate Analysis of Modes in LabVIEW

Early testing of the VITAMIN protocol on the USRPs showed that the VITAMIN protocol was working and that all modes had unique operating thresholds, that seemed to follow the theoretical order but at different E_s/N_0 values. In all 13 cases, early analysis suggested that all thresholds required higher SNR levels than the theoretical values from [18]. In order to identify why the performances levels are higher than the theoretical values it became necessary investigate the systems individually. Although the encoders and decoders acquired from the Jet Propulsion Laboratory were confirmed to meet the thresholds, the code was modified and used differently; thus the need to test the performances as a LabVIEW simulation without the USRPs.

To test the LabVIEW code, noise scaling operations, encoders, soft demodulates, and decoders were created in a separate LabVIEW VI, shown in Figure 5.1. These simulations are very similar to the simulations of Chapter 2. The CWER curves for modes 0, 7 ,8, 9, 11, 14, and 15 are plotted in Figure 5.2.

In all cases the observed threshold was less than the theoretical threshold. The difference was anywhere from 0.17 to 1.70. The number of simulations varied for the modes, modes 11 and 15 are the shortest in length resulting in more confident threshold observations, while the longer and more processing consuming BPSK modes have less confidence. In all cases, when performing a CWER test of 10,000 simulations at the theoretical threshold at least 2 errors were observed. From this observance shown in Table 5.1, it is concluded that there is an implementation loss in the LabVIEW code. Further investigation of the soft demodulators, noise estimation, and various decoder parameters will result in overall improvement. Since the performances are relatively close to the theoretical, additional improvements in the decoding will be deferred to suggestions for future work.

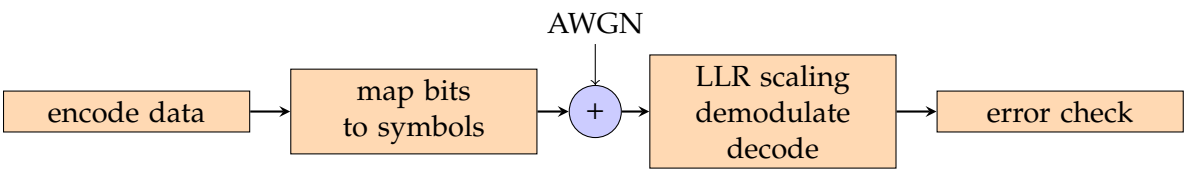

Figure 5.1. LabVIEW Software Only CWER Simulation Flow

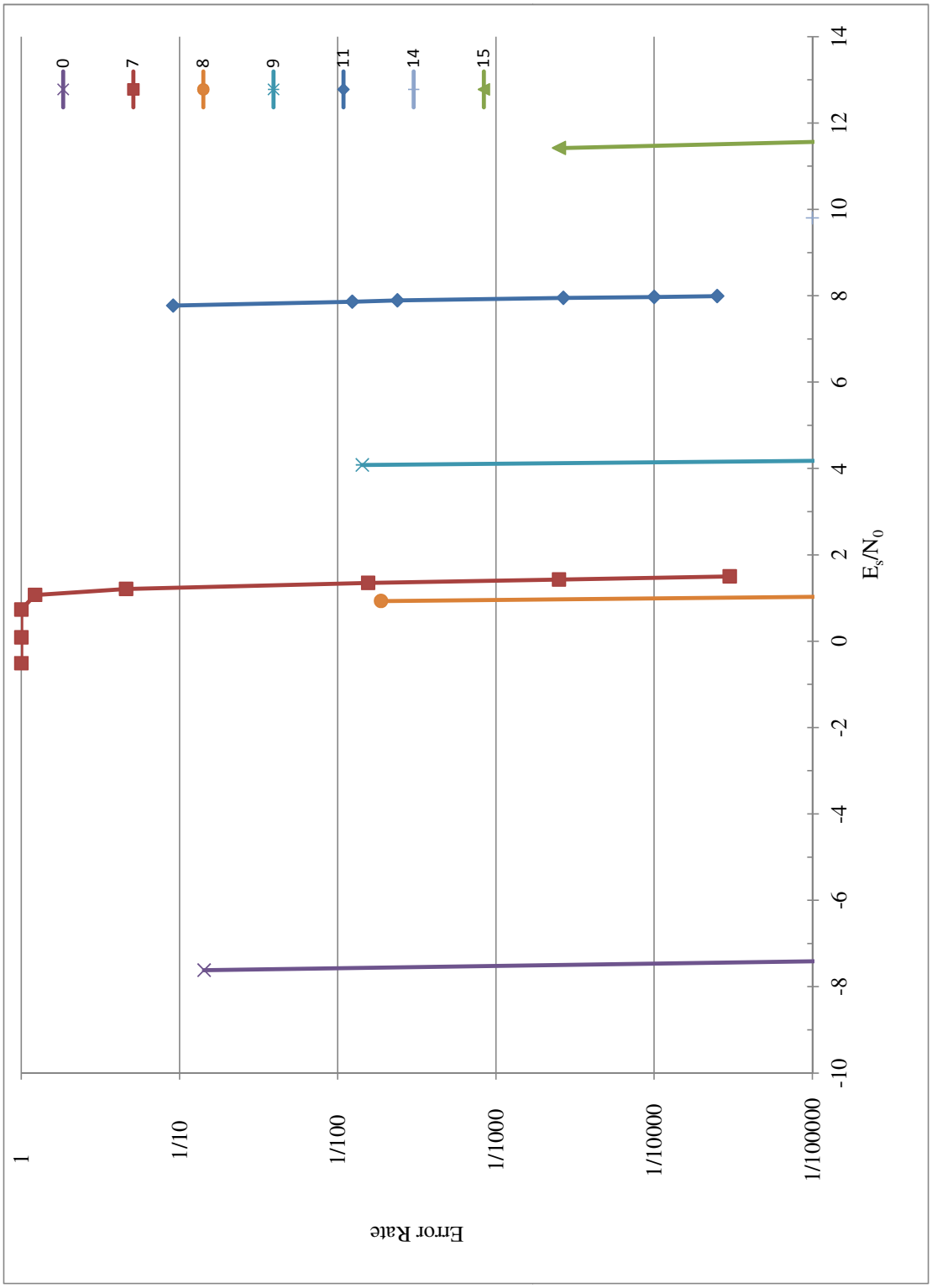

Figure 5.2. CWER Graph for Select Modes

Table 5.1. LabVIEW Software Only CWER

Mode	Name	Observed E_s/N_0 for CWER 10^{-4}	Theoretical E_s/N_0 for CWER 10^{-4}	Imp. Loss	Simulated Errors
0	BPSK Turbo 1/6	-7.4	-7.87	0.51	2
1	BPSK Turbo 1/4	-5.3	-5.84	0.55	2
4	QPSK Turbo 1/6	-3.8	-4.86	1.02	5
5	QPSK Turbo 1/4	-2.1	-2.83	0.80	5
6	QPSK Turbo 1/3	-0.7	-1.32	0.62	6
7	QPSK Turbo 1/2	1.5	1.07	0.44	4
8	QPSK AR4JA LDPC 1/2	1.1	0.93	0.17	10
9	QPSK AR4JA LDPC 2/3	4.4	3.00	1.40	3
10	QPSK AR4JA LDPC 4/5	6.3	4.75	1.54	3
11	QPSK C2 LDPC 7/8	8.0	6.27	1.70	16
12	8PSK AR4JA LDPC 1/2	4.4	4.04	0.27	4
13	8PSK AR4JA LDPC 2/3	7.7	6.71	1.03	7
14	8PSK AR4JA LDPC 4/5	9.8	8.94	0.86	4
15	8PSK C2 LDPC 7/8	11.6	10.79	0.81	12

5.2 Codeword Error Rate Analysis for USRP via RF

Codeword error rate simulations were performed to verify that the receiver was able to operate efficiently under noisy conditions. This test was performed by having the transmitter continuously transmit PLFs of mode type under constant Additive White Gaussian Noise. A PLF consisted of one Frame Marker and one encoded frame, thus making it one codeword per PLF. The receiver then continuously runs and records all PLFs that are successfully decoded. Simulations were all done at the same symbol rate of 50 kS/s. This symbol rate was found to be a the maximum speed that the desktop computer could support for the turbo codes, anything higher resulted too many packets building up in memory waiting to be decoded. Filling the memory up too fast resulted in simulations that would crash after only processing a couple hundred frames. Going to a slower symbol rate would have taken too much time; in the case of LDPC modes this extra time would have been unnecessary. Simulations would run anywhere from 1000-8000 codewords at a time. The number of simulations processed depends on the confidence level of the measurement. The confidence interval was calculated using the Monte Carlo simulation of n_{trails} with n_{errors}, this is easily done with the MATLAB "berconfint" function.

Additive White Gaussian Noise was inserted into the signal digitally at the transmitter through the MT AWGN LabVIEW function. The produced SNR (E_s/N_0) was verified through the spectrum analyzer, as seen in Figure 5.4. Since the USRP is an uncalibrated

device the actual bandwidth of the receiver and SNR received on the receiving end is not guaranteed to be the same as measured on the spectrum analyzer. To get an idea of the actual SNR, an SNR estimator was added to the Frame Marker detection subsystem of the LabVIEW receiver. Whenever a frame marker is received, the receiver has 256 noisy symbols and knowledge of what those 256 noisy symbols should be. By taking the difference between the received symbols and the known frame marker the noise power is known. Equation 5.1 equation gives the SNR of the frame marker,

$$SNR_{dB} = 20 \log_{10} \frac{RMS(y[i])}{RMS(n[i])}. \tag{5.1}$$

Over the simulation of thousands of packets the measured average SNR is calculated. This method always produced a SNR estimation that was lower than the spectrum measured SNR level. The results show spectrum and receiver SNR measurements and labeled respectively SNR-S or SNR-R.

Table 5.2 shows the results of the CWER simulations in LabVIEW. Figure 5.3 shows the CWER curve for mode 11, which is the best collection of simulation data due to required high SNR which resulted in only having errors due to noisy symbols and not receiver demodulation or phase issues.

Table 5.2. Performance under AWGN Observed with USRP Receiver for CWER 10^{-4}

Mode	Name	minimum SNR-S (E_s/N_0)	minimum SNR-R (E_s/N_0)
1	BPSK Turbo 1/4	8.0	7.7
10	QPSK LDPC 4/5	9.5	8.7
11	QPSK LDPC-C2 7/8	10.8	9.3
15	8PSK LDPC-C2 7/8	13.5	10.9

5.3 Simulation of CubeSat Pass

An intended application of the VITAMIN protocol is low orbit CubeSat communications, the first performance test was a real time satellite pass. The mode ordering was determined from the TMS program. Using the same hypothetical 600 km by 600 km 98 degree inclination CubeSat and and Fairbanks, AK groundstation in Chapter 3, a pass with maximum elevation of 64 degrees was selected. Table 5.3 describes all the parameters of the experiment.

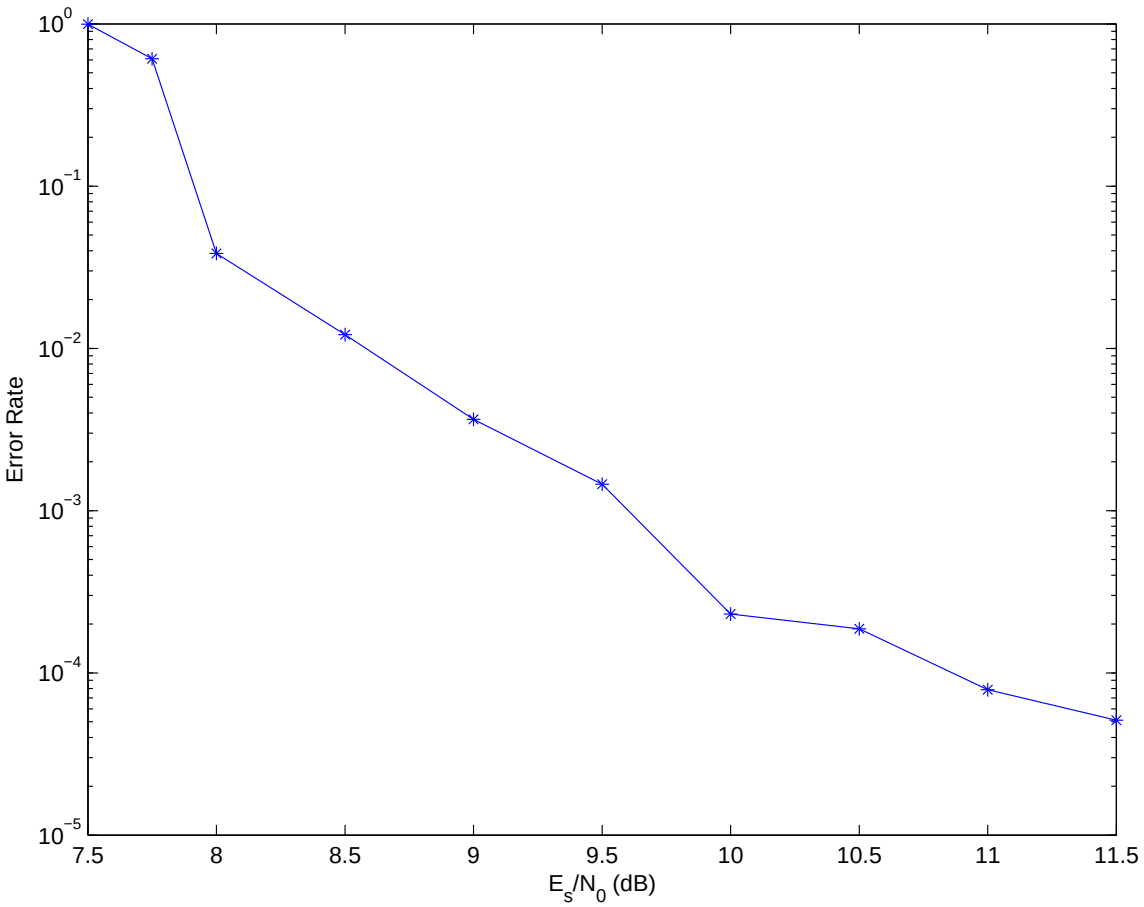

Figure 5.3. CWER curve for Mode 11

Table 5.3. VITAMIN Pass Simulation on USRP

Orbit	600 km by 600 km
Ground Station Location	Fairbanks, AK
Duration	0:12:49 or 769 seconds
Max Elevation	64 degrees
Frequency	2.2 GHz
TX Power	-30 dBm 1 μW
RX Power	-60 dBm or 1 nW
Modes Transmitted (order of 1st appearance)	4, 5, 7, 9, 12, 10, 11, 13, 14
3 dB Bandwidth	20 kHz
Symbol Rate	12.5 kS/s
Receiver IQ Data rate	250k
Filter Parameters	Root Raised Cosine, $\alpha = 0.5$, length 8
Physical Layer Frames	863
Frame Markers	864
AWGN	constant E_s/N_0 of 10 dB
Information Downlinked	1.44 MB

5.3.1 Process and Observations

Each desktop computer - USRP pair were setup to the above parameters. The MIMO cable was not connected. The signal was transmitted over coaxial cable with -30 dB of attenuation to create a received power less than -60 dBm, which is typical of CubeSats [24]. The receiver started with the run command, within 3 seconds the transmitter run command was issued.

The pass lasted just over 12 minutes and all packets were received. Although this simulation did not have changing noise, it proved the functionality of the receiver as it successfully detected various modes and was able to process variable coding and modulation types. Most importantly the receiver successfully recovered every information bit that was transmitted.

An interesting observation was the effects of decoding time. LDPC modes decode faster than the packets come in at 12.5 kS/s, turbo modes do not. Turbo codes are used when SNR is low, this results in the turbo modes occurring in the beginning and end of the pass. This was seen as the recovered packets were time delayed for the first 115 packets. Once LDPC encoding began, the receiver caught up to real time decoding. Likewise there was approximately 2 minutes of post processing to decode the trail end turbo codes. This delay is acceptable for most applications including data payload downlink.

This experiment was repeated over the air with 1 mW of output power at 2.2 GHz, but with the addition of the MIMO cable to correct for frequency and phase offset. There was a 6% packet loss due to phase ambiguity issues, which is discussed in subsection 4.6.6. Figure 5.4 shows the signal on a spectrum analyzer, the -30 dBm signal is 10 dB above the 500 kHz wide noise floor.

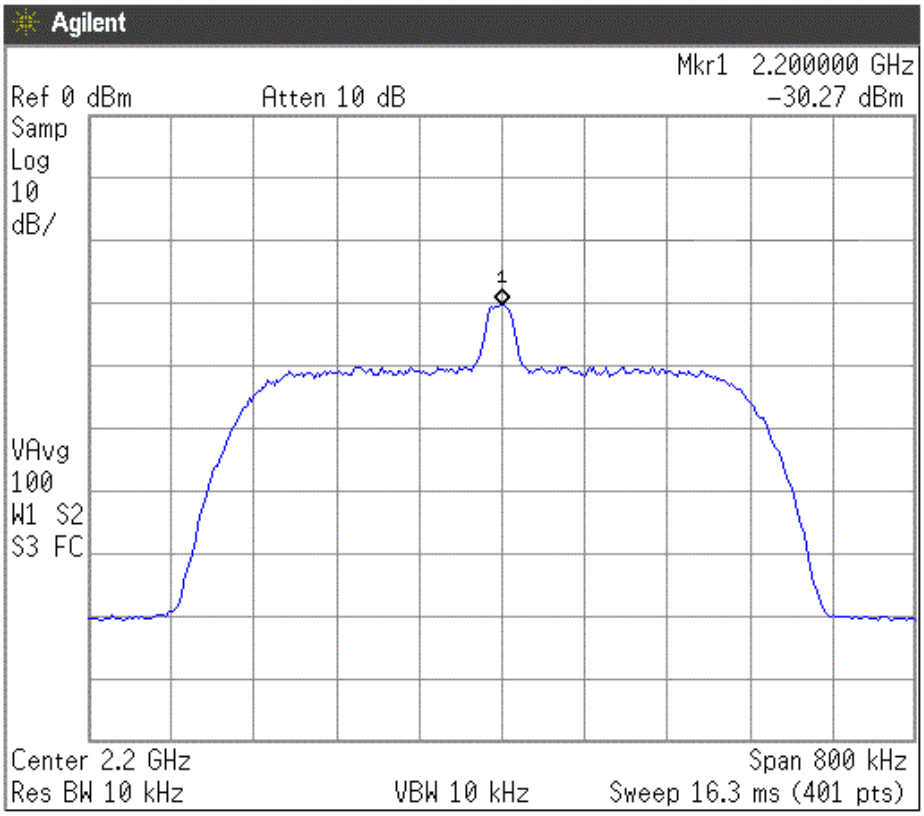

Figure 5.4. Spectrum of USRP TX Output with Noise

5.4 Simulation of Random VCM Mode

To further investigate the mode detection algorithms in the receiver, the transmitter was modified to encode with the selection of a random mode. The receiver was tested with 500 PLFs of varying code and modulations (modes 0-15), every frame was recovered. This shows that the receiver is able to switch from any mode or modulation, even transitions across the link budget, that would rarely if ever be necessary in real link budgets.

5.5 LabVIEW Performance Under Noise

The biggest issue with the LabVIEW receiver is operation under noisy conditions, particularly dealing with the demodulator. Ignoring the issues of mixing modulations in the "MT Demodulate VI", there are also issues with locking in the presence of noise. Without a stable lock, resolving phase, frequency, and symbol timing will produce codeword errors and the receiver fails. Frequency and phase offset between the receiver and transmitter were investigated by using the MIMO cable to share the transmitter reference frequency and clock with the receiver. Unfortunately this did not solve the issues of locking the demodulator under high noise.

LabVIEW documentation for the MT Demodulate function says the following for locking:

> Successful locking depends on many factors, including signal quality, modulation type, filtering parameters, and acquisition size. Locking also requires a fairly uniform distribution of symbols in the signal. The demodulator lock rate increases (and failures decrease) as the number of symbols demodulated increases. In general, you can expect to achieve a better than 95% lock when demodulating 10 M_{number} of symbols, where M is 2 bits per symbol.

Chapter 6
Final Recommended VITAMIN Protocol

6.1 Purpose

Variable Coded Modulation to Maximize Information is a VCM protocol that is a variation of CCSDS 131.2-B-1 [12]. VITAMIN's goal is to introduce an efficient coding and modulation solution able to support a wide range of information data rates at many SNR ranges. The main application is increasing the downlink payload of high data rate telemetry applications, like CubeSats in low Earth orbit.

6.2 Scope

VITAMIN's core modulation and coding modes are based off the combinations of 6 modulation types and three coding types. Turbo, AR4JA LDPC, and C2 LDPC codes were selected from NASA recommendations [14]. Only 24 modes are recommended, although the protocol encourages addition of other modes and in theory can support an unlimited number of modes. The stipulation for introducing additional modes is ensuring unique symbol lengths, see subsection 6.3.5. Table 6.1 lists the 24 recommended modes.

6.3 Data Handling

Figure 6.1 visually shows the flow of the data from the first stage of information bits to the final stage of modulated IQ pairs in the PLF.

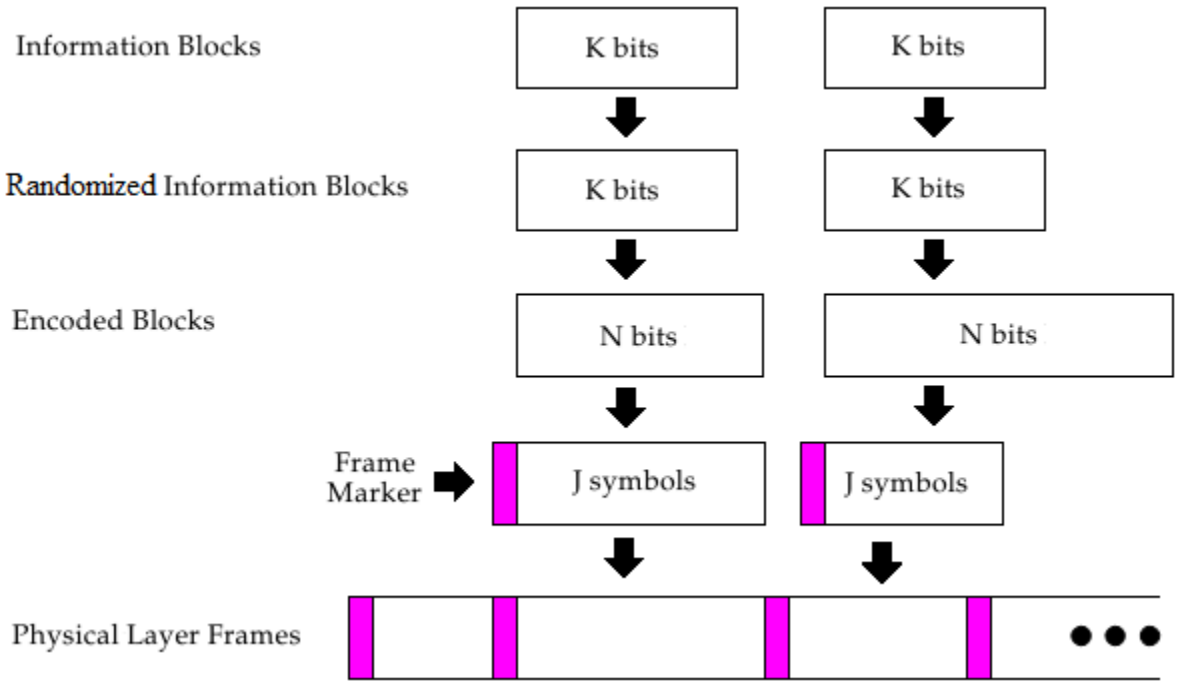

Figure 6.1. Stream Format at Different Stages of Processing

6.3.1 Signal Constellations

VITAMIN defines the signal constellations for the modulations identical to that of Figure 2 in [18]. Gray coding is required to achieve optimal results. The unsigned 8-bit 0,1 to signed 8-bit 1,-1 mapping is upheld throughout VITAMIN. Figure 4.12 visually shows the BPSK, QPSK, and 8PSK modulations.

6.3.2 Pseudo Randomization

To ensure proper operation, the data stream must be sufficiently random. The following polynomial describes the 255 pseudo-random sequence. The all 1 state is used in initialization,

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1. \tag{6.1}$$

A bit by bit exclusive-OR is applied to the entire data stream, the 255 bit sequence is simply repeated as necessary. The identical procedure is done on the receiving end to recover the data stream.

6.3.3 Frame Marker

The Frame Marker (FM) is generated using the Gold sequence using the following polynomials for the feedback loop:

$$g_1(x) = x^8 + x^6 + x^5 + x^4 + 1 \tag{6.2}$$

$$g_2(x) = x^8 + x^6 + x^5 + x^4 + x^3 + x + 1. \tag{6.3}$$

The complete generation process is described in section 5.3.2.2.1 of [12].

6.3.4 Physical Layer Frame

The Physical Layer Frame consists of one Frame Marker followed by one encoded frame. The process then repeats. An initial FM and final FM are required at the beginning and end of transmission. VITAMIN has 24 recommended modes which are listed in Table 6.1. Table 6.2 shows the 8 FEC that are used to make up the 24 modes. Additional modes can be added as needed, with the one requirement of uniqueness. In addition to all modes having unique encoded lengths, no mode can have a PLF length that is a multiple of another plus 256 symbols for every multiple. This property allows the receiver to differentiate between 2 PLFs of one mode and a single PLF of another mode.

6.3.5 Data Recovery Properties

Since the PLF only contains the FM and encoded data, VITAMIN highly recommends that the receiver use two FMs to determine the operating mode. Other possible techniques do exist, like parallel mode processing and trail and error processing. Having unique encoded symbol lengths as mentioned above subsection 6.3.4 and a constant symbol rate allow the receiver to deduce the PLF length by finding two consecutive FMs while buffering in between IQ pairs for slightly time delayed processing. It is also important to note that VITAMIN is flexible for adjusting the symbol rate in between transmission sessions, as everything scales to higher or lower symbol rates.

The concept of having no FDs in the protocol is a change from [12] and a new idea introduced in this thesis. The befits include better performance with low complexity. The ability to handle more than 32 modes, which is a limit of [12].

6.3.6 Mode Ordering

In applications where mode E_s/N_0 is described as a function of time, VITAMIN recommends a top down approach to determining mode ordering with regards to time, see subsection B.3.2. This thesis fully investigates mode ordering in Chapter 3.

Table 6.1. 24 VITAMIN Modes

	Modulation	Code	Rate	Info Bits per Symbol	Encoded Symbol Length	E_s/N_0 for CWER 10^{-4}
0	BPSK	Turbo	1/6	0.17	53520	-7.87
1	BPSK	Turbo	1/4	0.25	35680	-5.84
4	QPSK	Turbo	1/6	0.33	26760	-4.86
5	QPSK	Turbo	1/4	0.50	17840	-2.83
6	QPSK	Turbo	1/3	0.67	13380	-1.32
7	QPSK	Turbo	1/2	1.00	8920	1.07
8	QPSK	AR4JA LDPC	1/2	1.00	16384	0.93
9	QPSK	AR4JA LDPC	2/3	1.33	12288	3.00
10	QPSK	AR4JA LDPC	4/5	1.60	10240	4.75
11	QPSK	C2 LDPC	7/8	1.75	4080	6.27
12	8PSK	AR4JA LDPC	1/2	1.50	10923	4.04
14	8PSK	AR4JA LDPC	4/5	2.40	6827	8.94
15	8PSK	C2 LDPC	7/8	2.62	2720	10.79
16	16APSK	AR4JA LDPC	1/2	2.00	8192	6.33
17	16APSK	AR4JA LDPC	2/3	2.67	6144	8.98
18	16APSK	AR4JA LDPC	4/5	3.20	5120	11.19
19	16APSK	C2 LDPC	7/8	3.50	2040	12.94
20	32APSK	AR4JA LDPC	1/2	2.50	6554	8.98
21	32APSK	AR4JA LDPC	2/3	3.33	4916	11.63
22	32APSK	AR4JA LDPC	4/5	4.00	4096	13.90
23	32APSK	C2 LDPC	7/8	4.37	1632	15.73
24	64APSK	AR4JA LDPC	1/2	3.00	5462	10.97
26	64APSK	AR4JA LDPC	4/5	4.80	3414	16.61
27	64APSK	C2 LDPC	7/8	5.25	1360	18.60

Modes 2,3,13,25 are unassigned

Frame Length + 256 symbol FM is PLF length

Table 6.2. Forward Error Correcting Coding Types in VITAMIN

	Modes	Type	Rate	Length k	Encoded Length n
1	0,4	Turbo	1/6	8920	53520
2	1,5	Turbo	1/4	8920	35680
3	6	Turbo	1/3	8920	26760
4	7	Turbo	1/2	8920	17840
5	8,12,16,20,24	LDPC	1/2	16384	32768
6	9,17,21	LDPC	2/3	16384	24576
7	10,14,18,22,26	LDPC LDPC	4/5	16384	20480
8	11,15,19,23,27	LDPC-C2	223/255 (7/8)	7136	8160

6.4 Future Work

6.4.1 Bin-Packing

The Top Down algorithm was developed from this thesis and is proven to be an effective method for VCM mode ordering, but it can be improved upon. Digging deeper into bin-packing theory and other heuristic approaches could offer valuable insight. Only circular LEO orbits were analyzed, which present bell shaped SNR curves that are rather simple. Ground stations can have masking profiles, bind spots, and predictable interference which will add complexity to the SNR curve and require more flexible algorithms in the TMS program. SNR curves that have multiple peaks and sharper transitions will require an improved algorithm for VCM mode selection.

6.4.2 Fully Functional Software Defined Radio Ground Station

In order to obtain a fully functional software defined radio ground station several improvements are necessary. To support higher symbol rates and real time data gathering it is necessary to increase the speed of decoders. The LDPC codes are currently fast enough to support 10^9 symbols per second, while the turbo decoders only support speeds of 10^3 symbols per second. This can be done by removing initialization of certain parameters and pre-computing the lookup tables and matrices for the executable functions. There are also improvements in memory management and data passing that can be investigated in LabVIEW.

A final product will require further investigation of the codeword error rate performance, mainly the discrepancy between the LabVIEW results and theoretical performances.

There are alternative techniques for noise measurement and SNR estimation.

The VITAMIN protocol has several amplitude shift keying modulation types, which were not implemented due to added complexity. The USRP is documented as supporting this modulation type. The addition of these modulation types will greatly increase the overall throughput for applications that are able to operate with low noise levels.

Another way to increase the amount of data received is to attempt to recover any packet errors that may occur. Added complexity to the receiver will accomplish this. Currently a missed detection of a frame marker will result in two missed packets, but a smart receiver could correct this error. Re-examining received frames that are unsuccessfully decoded is worth the extra processing time. Simply researching the frame marker at another threshold level could result in recovered data. Another method would be trail and error decoding: each decoder will produce a valid flag if the packet is correctly recovered which could result in recovered data.

Further work is needed in investigating the benefits of a carrier signal and using coherent detection versus the current noncoherent detection system. The demodulators used in the LabVIEW receiver would benefit and possibly work as is if the incoming signal contained a carrier. This could result in a higher success rate of locking the demodulators. Research into other methods for coherent detection is advised. If a coherent method solves demodulation issues, analysis must be done to ensure the added complexity and power cost in the transmitter does not create more harm than good.

This thesis does not investigate the Doppler affect. Research into how this could affect performance under noise, frame marker detection, and overall throughput performance of VITAMIN is necessary as all moving satellites will have this phenomenon.

6.4.3 Develop a CubeSat Sized Transmitter

Although it has been researched that a VITAMIN transmitter can be implemented in the small and power constrained CubeSat with FPGAs, it would be very valuable to prototype this transmitter and measure power consumption. Analysis of power consumption versus time will be valuable information as CubeSats can have additional power resources at different times in the orbit, introducing the idea of preprocessing the encoding and symbol mapping steps and retaining maximum power for signal transmission.

Testing of the transmitter's ability to process the input data payload and function from an up-linked mode versus time input file created by the TMS program will bring forward

any issues with the pass prediction techniques and system architecture.

Acronym Index

SCCC Serial Concatened Convolutional Code 7

SNR signal-to-noise ratio 1

USRP Universal Software Radio Peripheral 35

VCM Variable Coded Modulation 3

VI File-extension: Variable Information 52

VITAMIN Variable Coded Modulation to Maximize Information 7

Bibliography

[1] I. Nason, J. Puig-Suari, and R. Twiggs, "Development of a family of picosatellite deployers based on the CubeSat standard," in *Aerospace Conference Proceedings, 2002. IEEE*, vol. 1, 2002. doi: 10.1109/AERO.2002.1036865 pp. 1–457–1–464 vol.1.

[2] L. David. (2004, Sept. 8) CubeSats: Tiny spacecraft, huge payoffs. Space.com. [Online]. Available: http://www.space.com/308-{C}ube{S}ats-tiny-spacecraft-huge-payoffs.html

[3] W. Stallings, *Data and Computer Communications*. Prentice Hall, 2011.

[4] M. Bernhardt. (2009, Jun. 12) Rapid terrestrial imaging CubeSat constellation. University of Washington Dept of Aeronautics. [Online]. Available: http://www.agi.com/downloads/partners/edu/UW_PDR_2009_paper.pdf

[5] SatelliteDish.com. (2011, Jul. 11) Large dish. [Online]. Available: http://www.satellitedish.com/page-6.htm

[6] J. Cutler and J. Mann, "Global ground station survey," in *CubeSat Developers Workshop*. Stanford University, 2008, pp. 1–15. [Online]. Available: http://gs.engin.umich.edu/documents/Mann_etal_2008.pdf

[7] M. Caffrey and J. Palmer, "Exploiting link dynamics in LEO-to-ground communications," in *CubeSat Developers Workshop*. San Luis Obispo, CA: Los Alamos National Laboratory, 2009, pp. 1–15. [Online]. Available: http://www.{C}ube{S}at.org/images/{C}ube{S}at/presentations/DevelopersWorkshop2009/4_Comm/2_Caffrey-Link_Dynamics.pdf

[8] S. Olivieri, "Modular FPGA-based software defined radio for CubeSats," Master's thesis, Worcester Polytechnic Institute, Worcester, MA, 2011. [Online]. Available: http://www.wpi.edu/Pubs/ETD/Available/etd-042711-091356/unrestricted/solivieri.pdf

[9] BusinessCom Networks. (2013, Nov. 11) Dvb-s2/acm services. [Online]. Available: http://www.bcsatellite.net/dvb-and-dvbs2-acm/

[10] J. Lee, "Richard Wesley Hamming: 1915-1998," in *Annals of the History of Computing, IEEE*, 1998. doi: 10.1109/MAHC.1998.667309 pp. 60–62.

[11] X. Qiu and K. Chawla, "On the performance of adaptive modulation in cellular systems," in *IEEE Trans. Commun.*, vol. 47, no. 6, 1999. doi: 10.1109/26.771345 p. 884 895.

[12] "Flexible advanced coding and modulation scheme for high rate telemetry applications," in *Recommendation for Space Data System Standards, CCSDS 131.2-B-1*. Washington, D.C.: CCSDS Blue Book Issue 1, Mar 2012. [Online]. Available: http://public.ccsds.org/publications/archive/131x2b1.pdf

[13] "TM synchronization and channel coding," in *Recommendation for Space Data System Standards, CCSDS 131.0-B-2*. Washington, D.C.: CCSDS Blue Book Issue 2, Aug 2011. [Online]. Available: http://public.ccsds.org/publications/archive/131x0b2ec1.pdf

[14] J. Hamkins, L. Deutsch, D. Divsalar, S. Dolinar, D. Lee, F. Stocklin, J. Wesdock, and C. Patel, "Formulation of forward error correction recommendations for future NASA space communications," in *IEEE Aerospace Conference*, Big Sky, MT, Mar. 2008. doi: 10.1109/AERO.2008.4526321 pp. 1–18.

[15] B. Sklar. (2008) Fundamentals of turbo codes. University of Massachusetts Lowell. Lowell, MA. [Online]. Available: http://faculty.uml.edu/jweitzen/16.548/classnotes/art_sklar3_turbocodes.pdf

[16] S. Moser and P. Chen, *A Students Guide to Coding and Information Theory*. Cambridge University, 2012.

[17] D. MacKay, *Information Theory, Inference & Learning Algorithms*. New York, NY, USA: Cambridge University Press, 2002. ISBN 0521642981

[18] J. Hamkins, "Performance of low-density parity-check coded modulation," in *IEEE Aerospace Conference*, Big Sky, MT, Mar. 2010. doi: 10.1109/AERO.2010.5446927 pp. 1–14.

[19] *Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications*, Digital Video Broadcasting Satellite Standard Std. ETSI EN 302 30, 2006.

[20] T. Sielicki, J. Hamkins, and D. Thorsen, "Variable coded modulation software simulation," in *IEEE Aerospace Conference*, Big Sky, MT, Mar. 2013. doi: 10.1109/AERO.2013.6497354 pp. 1–7.

[21] S. Sweep. Three dimensional bin-packing issues and solutions. University of Minnesota Morris. Morris, MN. [Online]. Available: http://micsymposium.org/mics_2004/Sweep.pdf

[22] P. Hoffman, *The Man Who Loved Only Numbers: the Story of Paul Erdos*. New York, NY: Hyperion, 1998.

[23] R. Lyons, *Understanding Digital Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 2010.

[24] D. Ichikawa. (2007, Feb. 23) CubeSat to ground communication and mobile modular ground station development. University of Hawaii at Manoa. Honolulu, HI. [Online]. Available: http://www.spacegrant.hawaii.edu/reports/16_SUM06-FA06/Ichikawa_Dylan_FA06.pdf

Appendix A
Thesis Source Code

A.1 Archive Files

All source code is attached with this thesis electronically through the Rasmuson Library at the University of Alaska - Fairbanks [`http://library.uaf.edu/`]. Alternatively I am available for questions and resources via email [tommy@sielicki.com] until the 22^{nd} century. Any proprietary code is removed.

A.1.1 MATLAB Simulation

Due to large number of proprietary functions source code is unavailable.

A.1.2 TMS Program

Java Eclipse export file "TMS.zip" contains source code and data files.

A.1.3 LabVIEW URSP Implementation

LabVIEW 2012 Project Solution archive "LABVIEW.zip" contains the VI files developed. Only transmission mode uncoded "-1" will work without the proprietary LDPC and Turbo encoder / decoder functions. Consider replacing the missing DLL files with alternative DLL files for other error correcting codes.

A.2 Code Highlights

Since LabVIEW is a visual dataflow programming language screenshots of select VIs are shown in Figure A.1, Figure A.2, Figure A.3, and Figure A.4.

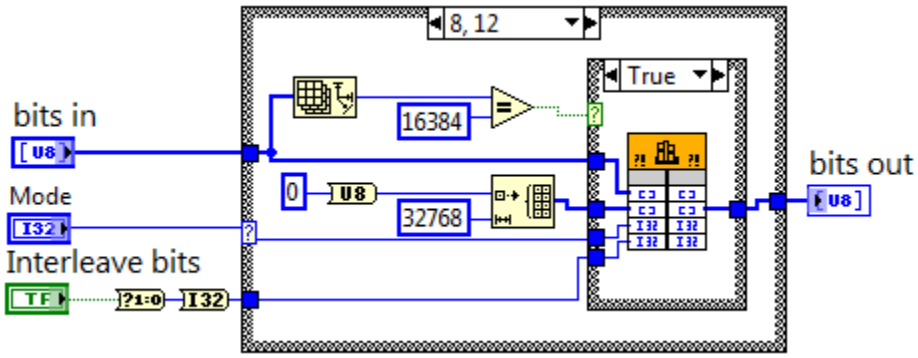

Figure A.1. Encoding Process - Interface with C DLL Function

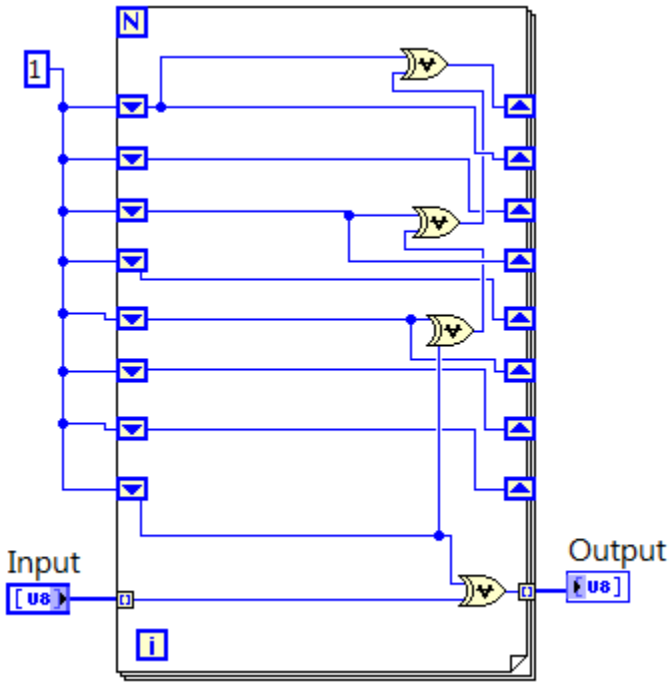

Figure A.2. Randomization Process Using Shift Registers

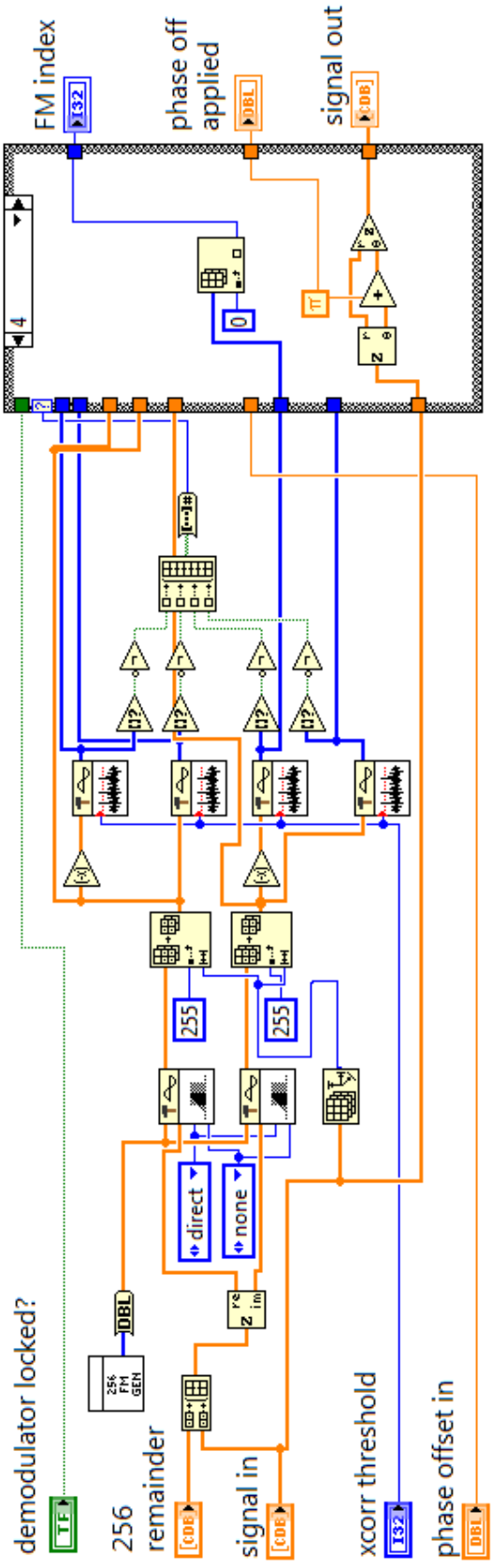

Figure A.3. Frame Marker Detection

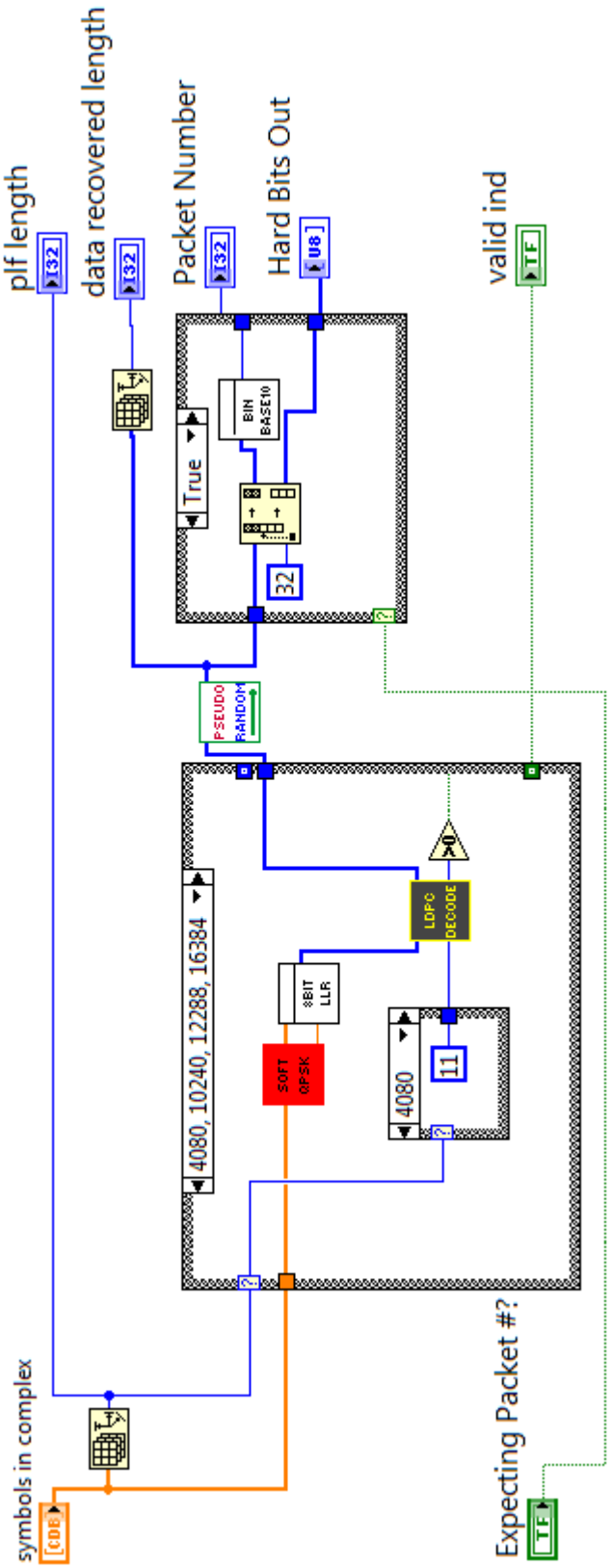

Figure A.4. PLF Processing: Complex Symbols to Hard Bits

Appendix B
Programs Specifications and User Guide

B.1 TMS Dependencies

- CSV file of the VCM modes to available to transmit from the spacecraft including the following details
  - Coding Type
  - Modulation Type
  - Coded length
  - FEC Rate (Information bits/sym)
  - Minimum E_s/N_0 need for communication with CWER of 10^{-4}
- Satellite Pass Report for a specific orbit and ground station (STK export)
  - Timestamp (minimum 1 second spacing)
  - E_s/N_0
  - Elevation
- Link Budget Constant (accounts for everything besides path loss)
- Length of the Frame Marker
- Number of consecutive frames to follow frame Marker
- Symbol Rate

B.2 TMS Outputs

- Shannon-Hartley Throughput (information bits)
- Best one mode per pass Throughput (percentage of Shannon Limit)
- VCM Throughput - using first fit algorithm
- VCM Throughput - using top down algorithm
- VCM Throughput - using random available algorithm
- Throughput - using traditional fixed rate communication system
- Mode switching commands file (for satellite uplink)
- VCM Modes used for data set including percentage of time used

B.3 TMS Bin-Packing Approaches

Using the classical Bin-Packing approaches to create programming methods for mode selection several techniques were created and compared.

B.3.1 First Choice Selection

1. If E_s/N_0 at t_0 < Minimum E_s/N_0 threshold for all modes, $t_0 = t_0 + 1$, repeat step 1, if no more data points exist communication ends, else continue
2. Check which modes can operate at E_s/N_0 at time t_0
3. Select the mode with the highest information capacity, called ”mode candidate”
4. Knowing the length of ”mode candidate” and t_0 determine the time PLF ends, t_e, then check which modes can operate at t_e
5. If mode candidate can operate at t_e, finalize ”mode candidate”, set t_0 to t_e and return to step 2.
6. If mode candidate cannot operate at t_e, return step 2 using next best mode, If no ”candidate mode” works, end communications.

B.3.2 Top Down Selection

1. Determine time at which E_s/N_0 is maximum for entire pass, called t_{max}
2. Determine which modes can operate at t_{max}, select mode with greatest information capacity called ”mode candidate”.
3. Determine time at which ”mode candidate” is first operable, called t_0
4. Determine t_e from mode candidate and check if mode candidate is operable at t_e, if not repeat step 2 but select next best mode.
5. Increase t_0 and t_e by one increment (default 1 second) until mode is not operable at either $t_0 or t_e$, then finalize mode candidate and time placement.
6. Solve from t_e to $t_{endofpass}$ using ”First Choice Selection” technique.
7. Solve from t_0 to $t_{startofpass}$ using ”First Choice Selection”, working backwards through time.

B.3.3 Random Solution

1. If E_s/N_0 at t_0 < Minimum E_s/N_0 threshold for all modes, $t_0 = t_0 + 1$, repeat step 1, if no more data points exist communication ends, else continue

2. Check which modes can operate at E_s/N_0 at time t_0
3. Select a mode from the list randomly, called "mode candidate"
4. Knowing the length of "mode candidate" and t_0 determine the time PLF ends, t_e, then check which modes can operate at t_e
5. If mode candidate can operate at t_e, finalize "mode candidate", set t_0 to t_e and return to step 2.
6. If mode candidate cannot operate at t_e, return step 2 using another random mode, If no "candidate mode" works, end communications.

B.3.4 One Mode Solution

1. Iterate through all modes
2. Determine time t_0 at which mode is operable
3. Determine the time at which mode is not longer operable t_e, and determine how many whole PLFs can fit in the time period $t_e - t_0$.
4. Calculate throughput and record
5. Return the result of the mode with the highest throughput

B.3.5 Fixed Mode Solution

1. Determine time t_0 at which mode is operable
2. Determine the time at which mode is not longer operable t_e, and determine how many whole PLFs can fit in the time period $t_e - t_0$.
3. Calculate throughput and record

B.4 LabVIEW Transmitter/Receiver User Guide

B.4.1 User Variables

The hardware simulation has many configurable user parameters. All inputs are easy to modify on the graphical user interface of the LabVIEW VI file. The transmitter needs to specify the transmission modes, this is done using an array listing the modes to be

transmitted in order from start to finish. The transmitter also is configurable to loop the modes 1 to N times, where a negative input will cause the transmitter to loop infinitely. By default the transmitted data is random, dictated by a seed. The user has the ability to insert a frame number into the data packet that can be extracted by the receiver, 2^{32} frame numbers are supported. Another input is for the frame marker detection function in the receiver, this is an adjustable threshold value that can range from 0-256. Pulse shaping is supports three types: Root Raised Cosine, Raised Cosine, and Gaussian. The user must specify *apha* and the filter lengths. The remaining user inputs deal mainly with the USRP itself: carrier frequency, symbol rate, IQ rate of the radio to pc, oversampling factor, antenna port, IP address of the USRP, and TX/RX gain.